An Automated End-to-End Lecture Capture and Broadcasting System

Cha Zhang, Yong Rui, Jim Crawford and Li-wei He
Microsoft Research

Remote viewing of lectures presented to a live audience is becoming increasingly popular. At the same time, the lectures can be recorded for subsequent on-demand viewing over the Internet. Providing such services, however, is often prohibitive due to the labor-intensive cost of capturing and pre/post-processing. This paper presents a complete automated end-to-end system that supports capturing, broadcasting, viewing, archiving and searching of presentations. Specifically, we describe a system architecture that minimizes the pre- and post-production time, and a fully automated lecture capture system called *iCam2* that synchronously captures all contents of the lecture, including audio, video, and presentation material. No staff is needed during lecture capture and broadcasting, so the operational cost of the system is negligible. The system has been used on a daily basis for more than 4 years, during which 522 lectures have been captured. These lectures have been viewed over 20,000 times.

Categories and Subject Descriptors: H.4.0 [**Information Systems Applications**]: General

General Terms: Lecture broadcasting

Additional Key Words and Phrases: Automated lecture capture, live/on-demand broadcasting

1. INTRODUCTION

Live/on-demand Internet broadcasting of lectures in the workplace, at conferences and in educational settings has attracted more and more interest due to improvements in network bandwidth, computer performance, and compression technologies. Many corporations make seminars and training sessions available for employees who cannot attend a live presentation [He et al. 2001; Steinmetz and Kienzle 2001]. Many conferences recorded their presentations and made them available for on-demand replay (e.g., SIGGRAPH[1] and NOSSDAV[2]). Many universities also make lectures available online for both regular and distance education, such as Stanford University[3] and Columbia University[4]. The UK Open University was the first to

[1]http://terra.cs.nps.navy.mil/DistanceEducation/online.siggraph.org/
[2]http://bmrc.berkeley.edu/research/nossdav05/
[3]http://scpd.stanford.edu/scpd/students/onlineclass.htm
[4]http://www.cvn.columbia.edu/

Contact author's address: Cha Zhang, Microsoft Research, One Microsoft Way, Redmond, WA 98052. Email: chazhang@microsoft.com.

webcast live and online degree ceremonies [Scott and Mason 2001], which received considerable attention.

Although online viewing provides a convenient way for people to watch lectures at a more convenient time and location, the cost of providing such services can be prohibitive. Rowe et al. [Rowe et al. 2001] and Rui et al. [Rui et al. 2001] studied the cost issues in their respective papers, and summarized the cost into two major parts: 1) fixed cost, which includes computer servers, microphones and cameras; 2) recurring labor cost, which includes pre-lecture activities (e.g., setting up the equipment), during-lecture activities (e.g., controlling cameras to track presenters and the audience and switching between cameras), and post-lecture activities (e.g., post the lecture to a web site).

Fixed cost is a one-time investment. Improvements in technology has seen a rapid decline in this expense over the past decade. Labor cost, on the other hand, occurs every lecture and has not decreased over time. Recently, there have been several efforts to build automated/intelligent capturing systems to reduce the labor cost [Liu and Kender 2004]. Examples include the AutoAuditorium system [Bianchi 1998; 2004], the "Classroom 2000" project [Abowd 1999] at Georgia Tech, the Cornell Lecture Browser [Mukhopadhyay and Smith 1999], the Berkeley Internet Broadcasting System (BIBS) [Rowe et al. 2001], the IBM e-Seminar lecture recording and distribution system [Steinmetz and Kienzle 2001], the Microsoft *iCam* system [Rui et al. 2001; Rui et al. 2004] and the University of Toronto ePresence system [Baecker 2003]. These systems have focused on different aspects of "automation". For example, some systems automated camera control and tracking of presenters and audience [Bianchi 1998; Rui et al. 2001; Onishi and Fukunaga 2004; Gleicher and Masanz 2000]; others automated digital ink and electronic slide capture [Abowd 1999; Wang et al. 2003]; yet others focused on automated broadcasting [Machnicki 2002; Baecker 2003]. Nevertheless, they all met their own application goal by following a set of design principles derived from what is available and what is necessary to reduce the cost of production.

This paper presents a complete end-to-end system, called Microsoft Research LecCasting System (MSRLCS), that is fully automated and has been used on a daily basis in our organization for over 4 years. The MSRLCS system was designed to meet a few stringent requirements raised in practice:

—**Support live broadcasting.** The system has to support live broadcasting so that remote users can watch the lectures live. It saves travel time (without live broadcasting some people in our organization could easily spend over half an hour round trip to the seminar room), gives users more flexibility when to start and stop watching the lecture, and allows people to multi-task if they like.

—**Short post-production delay.** Shortly after the talk, users may want to watch the lectures on-demand, hence a short post-production delay is a necessity.

—**On-the-fly slide capture.** Many of our speakers are external. It is often very difficult to get their slides before or after the talk, thus we need to develop a mechanism to grab the slides during the lecture.

—**Passive.** Since it is usually the first time for most external speakers to use our system, they have little time to learn and adapt to the system. The system needs to be truly passive, and pose no restrictions on the speakers' behavior (e.g.,

asking the speaker to stay within the field of view of a fixed camera).

—**Fully automated.** As for the system administrators who run and maintain the system, we want to minimize their work by making the lecture capturing and broadcasting processes fully automated. Such automation requires a carefully designed system architecture that needs no human interaction during the broadcasting. At the same time, the quality of the production matters, so techniques are required to track the speaker and the audience and to switch between different cameras and views in the lecture room.

—**Generalizable/portable.** The system must be generalizable so that it can be installed in conference rooms with different configurations.

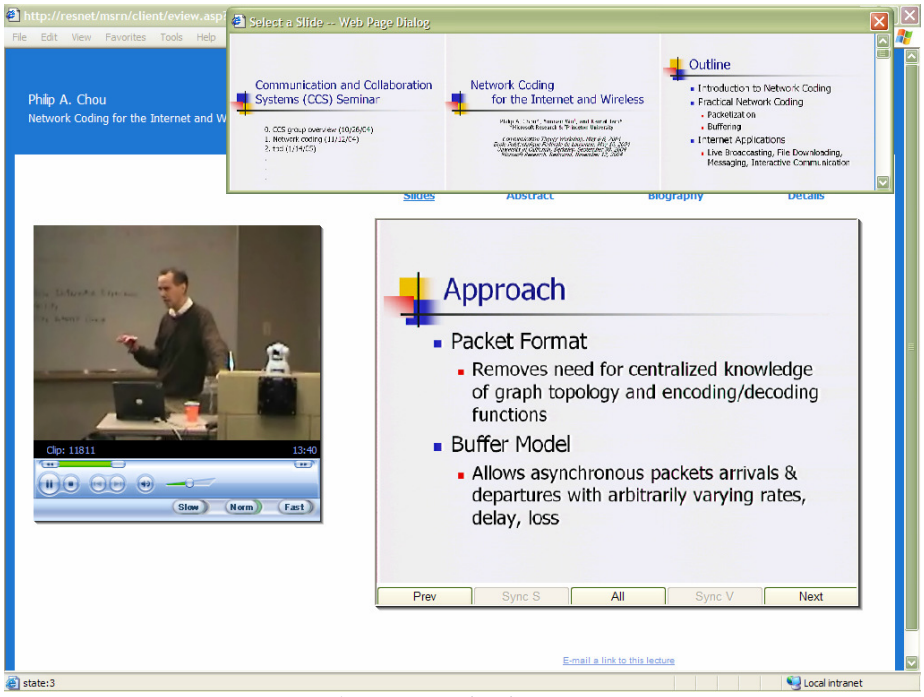
With these requirements in mind, we have carefully designed the architecture of the MSRLCS system so that it has minimum pre- or post-production and requires no operators during the entire broadcast. As soon as the speaker walks into the lecture room to prepare for his/her presentation, the system administrator starts the capturing and live broadcasting process with a single click of a button. Remote users can immediately see what is happening in the lecture room with a web browser. Figure 1 shows the web interfaces for lecture browsing on the remote client. Figure 1(a) shows the talk schedule. It has a calendar in the top left corner. If there are lectures available, the corresponding date appears in bold face. A user can click on any date to retrieve detailed information about the talks available on that day, including the title, abstract, and speaker biography. For a recorded lecture, "Available" will appear under the talk title. The user can view the lecture on-demand by following this link. A live talk will show "In progress" under the title. When the user clicks on the link, the live viewing display is presented. The user can use keywords to retrieve a talk in which he/she is interested using the text-based search capability under the calendar.

Figure 1(b) is the interface for watching a lecture. The video steam is displayed on the left, and the slides are displayed on the right. If the lecture is live, the user can watch the current video and slides synchronously, similar to watching a live event broadcast by a TV station. If the lecture was recorded before, the user will have more flexibility. He/she can browse all the slides used in the talk (e.g., the floating window at the top of Figure 1(b)) by hitting the "All" button. Double clicking a thumbnail slide image will show the corresponding slide in the slide region. The user can also move the slider of the embedded Windows Media Player and jump to any place in the audio/video (AV) stream[5]. The "Sync V" button allows audio/video to synchronize with slides, and the "Sync S" button synchronizes slides to audio/video. Note the current system does not support slide browsing and re-synchronization during a live session.

The contents of the lecture, including audio, video, and visual aids such as slides, are captured synchronously with our automated lecture content capturing system called *iCam2*. The system is based on the *iCam* system [Rui et al. 2001], and focuses on generalizability while improving the overall performance of the capture. It captures high quality audio and generates professional-looking edited videos by switching between multiple camera views according to cinematographic rules. Visual aids

[5]This feature was first implemented by Eloquent. http://www.eloquent.com/

(a) Talk schedule page



(b) Lecture viewing page

Fig. 1. Live viewing web interface.

and slides are synchronously captured for easy browsing and re-synchronization during on-demand viewing. Similar to *iCam*, the speaker can walk around during the presentation, and the system will automatically track him/her and any audience member who asks questions. Shortly after the lecture ends (i.e., less than one minute), the recorded lecture is available for on-demand viewing. The MSRLCS system was deployed in June 2001. By mid-August 2005, 522 lectures had been captured with 20,383 viewing sessions.

The contribution of this paper is two-fold. First, we present a complete end-to-end lecture capture and broadcasting system, which has been extensively used in the past few years. The design and implementation of the system are discussed in detail, making it clear how minimum pre- or post-production time is achieved. Second, we introduce *iCam2*, which is our second generation automated lecture content capturing system. *iCam2* shares many ideas with the previous *iCam* system [Rui et al. 2001], but has a number of notable improvements:

—Instead of using video cameras to capture the visual aids in *iCam*, we have an improved solution using a video capture card, which provides higher resolution images. Visual aids are synchronized with the AV stream by embedding commands into the stream signaling when a slide should be displayed.

—Use of a microphone array to improve capture of audience questions via a sound source localization and beamforming algorithm. This process also provides input to the camera switching algorithm.

—Development of a new algorithm to mix audio from different sources that are not strictly synchronized (e.g., audio from a wireless microphone used by the speaker, the microphone array, and audio output from the presentation laptop).

—For speaker tracking, we replaced two cameras (i.e., one static wide-angle view and one pan/tilt/zoom (PTZ) for speaker close-ups) used in the *iCam* system with a single PTZ camera, and developed a hybrid tracking algorithm that achieves similar tracking performance. The new hardware configuration also enhances the portability of the system.

—Support of a scripting language so that production rules (e.g., camera switching) can be customized for different room configurations and production styles.

Some of the technical components have been discussed in earlier publications. For instance, hybrid tracking was discussed in [Zhang et al. 2005], sound source localization and beamforming with a microphone array were described in [Rui and Florencio 2004; Tashev and Malvar 2005], and a preliminary version of the scripting language was presented in [Wallick et al. 2004]. In this paper, we will focus on *iCam2* and refer readers to these published papers for more technical details on the other aspects of the system.

In the rest of the paper, we first describe related work in Section 2. The design principles and system architecture are presented in Sections 3 and 4. The *iCam2* system for automated content production is detailed in Section 5. Section 6 describes the implementation of our system. System usage statistics and conclusions are given in Section 7 and 8, respectively.

2. RELATED WORK

As the name implies, a lecture capturing and broadcasting system has two major aspects — capturing and broadcasting. An ideal system should automate both aspects. This section reviews existing approaches on automated content capturing and automated broadcasting separately, although many systems have made progress on both aspects.

2.1 Automated content capturing

The contents of a lecture usually refer to the audio and video of the speaker and the audience, and visual aids such as slides or transparencies used by the speaker to facilitate the talk. It is also important to make sure that the captured material is well synchronized, so that it can be presented to remote clients in an organized fashion. The following paragraphs discuss prior work capturing audio, slides, and video.

2.1.1 *Audio.* Audio contains very important information in video conferencing [Finn et al. 1977]. A typical audio capturing system, such as the one used in the STREAMS system [Cruz and Hill 1994], includes a wireless speaker microphone and a set of audience microphones distributed in the lecture room. The signals from all microphones are mixed using an analog mixer. Although some analog mixers have built-in echo cancellation and noise suppression, most mixers require a human operator to adjust the gain for each channel during the lecture. For instance, when the speaker talks, the microphones capturing the audience should be muted to remove unnecessary noises. On the other hand, when a question is raised from the audience, the human operator needs to figure out the closest microphone and increase its volume. The AutoAuditorium system tackled this problem using a special "Mic Ducker" circuit that selectively muted the audience microphones, although that required additional hardware. Another constraint is that distributed microphones need to be wired to the mixer, which can be expensive to set up.

The MSRLCS system uses a custom-built eight-element microphone array to capture audience questions. A software-based algorithm performs identification and source localization of audience sound and mixes it with speaker sound captured by a wireless microphone. This solution demands less wiring, and is necessary because the audio signals from the microphone array and the wireless microphone are not synchronized due to different computer interfaces (e.g., the microphone array has a USB interface while the wireless microphone signal is captured with an audio capture card), which could result in strong echo if not treated carefully. The algorithm and the hardware used are described in Section 5.1.

2.1.2 *Visual aids.* Visual aids can be obtained off-line or captured on-the-fly. In the Classroom 2000 [Abowd 1999] system, speakers must load their presentations into the system before class and teach using electronic whiteboards. The Cornell Lecture Browser [Mukhopadhyay and Smith 1999] used automated post-production to synchronize audio and video captured during the lecture with slides acquired from the speaker afterwards. Unfortunately, many speakers are not willing to provide a copy of their slides. Amir et al. used a video camera to capture the slides and adapted a shot boundary detection algorithm to identify important slides [Amir

et al. 2004]. The ePresence system [Baecker 2003] got around the problem by an operator-initiated trigger which grabbed a scan converted representation of the data projector's output. The University of Minnesota uses a still image camera to capture whiteboard and projected images every couple of seconds. Selected images are synchronized with captured audio and video manually in post-production.

Converting the slides (i.e., an RGB signal) to an NTSC signal either by using a scan converter or by pointing a camera at the projected slides simplifies the capture process, but leads to problems with visual quality since the slides often contain too much information to be accurately captured and encoded as streaming video, typically a CIF image. A few existing systems have used RGB capture to overcome the problem. For instance, the IBM e-Seminar system [Steinmetz and Kienzle 2001] used the Datapath RBG Vision board for slide capture, and the NOSSDAV'05 capture experiment [Rowe and Casalaina 2006] used the NCast Telepresenter M3[6]. RGB capture produces better quality images, but may use too much bandwidth since the captured images can have significant duplication. In addition, picture in picture production is often required in order to show the presenter and the slides simultaneously.

The visual aids are captured in the MSRLCS system using an RGB capture card. An automated slide change detection algorithm is developed to avoid image duplication, hence no manual operation is needed during or after the presentation. In addition, slides and video are presented in separate windows, as shown in Figure 1(b). Slides, audio, and video are always synchronized via embedded scripts in the live stream. To handle animations, video and demonstrations produced on the presentation computer, the MSRLCS system currently uses a camera pointing to the projected region. The output video will be shown in the video window in Figure 1(b). More details will be described in Section 5.2.

2.1.3 *Video.* While audio conveys the major contents of the talk, it is video that makes a talk engaging [Tang and Issacs 1993]. A large amount of effort has been dedicated to video content production in the literature.

Hiring professional videographers usually offers the highest visually appealing quality, however the labor cost is often prohibitive. For low-cost systems, using a fixed camera is feasible and convenient, although it has limitations. For instance, if we use a fixed camera to capture the speaker, depending on the field of view of the camera, we may get a low resolution speaker shot or lose the speaker from time to time when he/she walks out of the field of view. Recently, people have used wide-angle high resolution cameras or PTZ cameras to capture the lecturer. Other lecture capture systems have used technologies for automated speaker and audience tracking, either online [Bianchi 1998; Rui et al. 2001; Onishi and Fukunaga 2004] or offline [Mukhopadhyay and Smith 1999; Heck et al. 2006].

Another common practice is to use multiple cameras to capture the scene, and (optionally) process the data afterward or switch between cameras on-the-fly. Several systems including STREAMS [Cruz and Hill 1994], Access Grid[7], and ConferenceXP, gave remote clients the flexibility to choose which stream to watch.

[6]http://www.ncast.com/
[7]http://www.accessgrid.org/

While providing all the streams to the users gives them flexibility, it increases the burden on the server bandwidth, and sometimes it can be distracting to the remote client. Several groups have automated the decision about which camera or view to present either during the live broadcast or afterwards during post-production. Generally such a process is guided by a set of rules either suggested by professionals [Rui et al. 2001] or by mining professionally produced videos [Matsuo et al. 2002]. The AutoAuditorium system [Bianchi 1998; 2004] automates the entire lecture capturing process very effectively. User may start and stop the lecture recording with the touch of one button. It is used in IBM's e-Seminar system [Steinmetz and Kienzle 2001]. Machnicki developed a system used in the Berkeley MIG Seminar that automates tasks such as control of recording equipment, stream broadcasting, camera control, and content decisions such as which camera view to broadcast during the live broadcast [Machnicki 2002]. The camera switching heuristics are mostly time-based. A question monitor service based on audio analysis was used to detect questions from the audience and switch the webcast to show the audience member asking the question.

Our solution to video capture is the *iCam2* system described in Section 5.3. Two cameras, one for the speaker and the other for the audience, are used to capture the lecture. A hybrid speaker tracking algorithm is developed using the PTZ camera at the back of the room (Section 5.3.1). The audience camera is guided by the microphone array for prompt response to audience questions. A simple scripting language was developed to specify camera transition rules in a nested finite state machine. This approach is in contrast to the previous *iCam* system [Rui et al. 2004], where transition rules were predefined and hard coded in the system.

2.2 Automated live/on-demand broadcasting

Few systems support live and on-demand lecture broadcasting on a daily basis, among which the BIBS [Rowe et al. 2001] system at UC Berkeley, the ePresence [Baecker 2003] system at University of Toronto, and the e-Seminar [Steinmetz and Kienzle 2001] system at IBM are the most representative.

BIBS has been adopted as an integral part of the university's course delivery infrastructure, and webcasts approximately 15 classes each semester[8]. The system was designed to require as few people as possible to operate.

The ePresence system is an open source system for lecture capture, archive and broadcasting. It is scalable, interactive, and able to support presenters and engage remote audiences with rich media. Due to its ambitious goals, ePresence was not designed to be fully automated – a moderator is required to coordinate the interactivity between remote users and speaker.

The e-Seminar lecture recording and distribution system used at IBM Research allows employees to access video and slides of talks, seminars, presentations and other events at any IBM-Research campus worldwide. It can handle broad spectrum of tasks, from scheduling to distribution and user feedback. The system uses AutoAuditorium for live production and capture. It also has special emphasis on automation during production and post-production of the recorded materials. For example, it performs camera-switching and uses special effects such as picture in

[8]http://webcast.berkeley.edu/

picture automatically.

There are a number of commercial lecture capture systems available today, such as the Anystream Apreso Classroom[9], Horizon Wimba[10], NCast Telepresenter[11] and Tegrity[12]. These commercial systems have different focuses and strengths. For example, the Apreso Classroom supports fully automated lecture capturing and web publishing, and features such as scheduled start and stop of the capturing process, synchronized audio and visual aids, and automated web publishing. Horizon Wimba's Course Genie is a course authoring tool for online course publishing. NCast Telepresentor is an embedded computer that captures audio and RGB signals and produces an mp4 file that can be played by a streaming video player. Tegrity is capable of supporting a variety of teaching styles and tools for lecture capture, storage and indexing. Nevertheless, none of the above systems capture audio/video with synchronized slides from the RGB signal. They also do not support automated camera switching during live production, which is important for creating attractive lecture materials.

This paper presents a live/on-demand broadcasting system that is fully automated and captures rich contents including audio, video and synchronized slides. As detailed later, the system architecture was carefully designed to minimize pre- and post-production time.

3. DESIGN PRINCIPLES

The general design principles of our system are derived from past experiences and grounded in results from the video communications literature such as [He et al. 2001; Baecker 2003]. The emphasis, however, is on automation. We want to minimize the work performed by humans, so that the system can run by itself on a daily basis, with little or no operational cost. This section highlights principles we believe are critical to our system. These principles are closely related to the requirements discussed in the introduction.

[P1] The system is passive. We would like the speaker to behave normally during the talk, thus we do not impose any restrictions on him or her. The only device the speaker needs to wear is a wireless clip-on microphone.

[P2] The system has no pre- or post-production. For instance, we do not require the speaker to give us their slides/transparencies for pre-processing. After the presentation, no post-production such as slide integration is needed. The lecture is immediately available for on-demand viewing.

[P3] The system captures synchronized high resolution visual aids. Such synchronization is done on-the-fly during the lecture. Both the live and the on-demand viewer can watch them synchronously with the audio/video stream of the lecture.

[P4] The system captures audio and video of the lecture automatically. A new version of the *iCam* system, called *iCam2*, has been developed that greatly enhances portability. In both generations, no videographer or moderator is needed

[9]http://www.apreso.com/
[10]http://www.horizonwimba.com/
[11]http://www.ncast.com/
[12]http://www.tegrity.com/

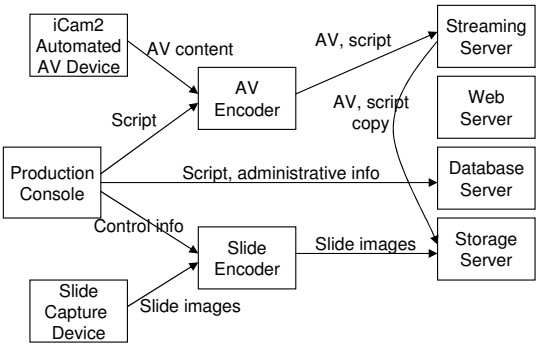


Fig. 2. System architecture of our system.

during the lecture.

[P5] The system needs to be portable across various room configurations. We select network PTZ cameras to capture the video as they require minimum amount of wiring. A scripting language is developed to make the cinematographic rules adaptive to a different number of cameras, camera locations, and room sizes.

[P6] The system allows the remote clients to view the lecture at their own pace. Metadata are sent along with the audio/video stream from the server. During on-demand viewing, the users can flip through the slides or jump around the audio/video stream. At any instance they can re-synchronize the audio/video with slides, or vice versa.

In the following text, we will annotate these design principles whenever applicable with the indexes [P1], [P2], etc.

4. SYSTEM ARCHITECTURE FOR AUTOMATED BROADCASTING

The MSRLCS system allows users to view lectures either live or on-demand through a web browser. Audio/video streams and slide images are synchronized but provided by multiple servers. In this section, we will first introduce the system architecture, then explain the data flow during live and on-demand broadcasting.

4.1 The capturing diagram

Figure 2 shows the architecture diagram of the MSRLCS system when capturing a lecture. The *iCam2* automated audio/video (AV) device and the slide capture device are the content providers, which will be discussed in detail in Section 5. The production console coordinates the AV and slide capturing processes. When a slide change is detected, the production console sends commands to the slide encoder to capture an image of the slide. At the same time, it embeds a script command into the AV stream, which tells the remote client to update the slide image. Hence, the AV stream and slides are always synchronized on-the-fly [P3]. The AV encoder process encodes audio, video, and the script information into a single live stream, and forwards it to the streaming server for live broadcasting. The slide encoder compresses slide images and sends them to a storage server. During

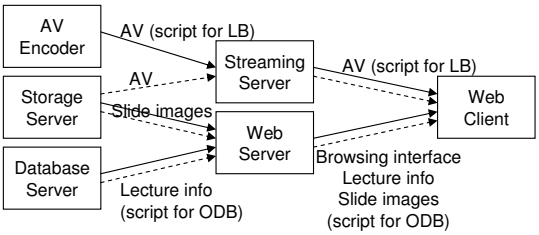


Fig. 3. Diagram for live broadcasting (LB) and on-demand broadcasting (ODB). The solid arrows are data flow for live broadcasting, and the dash arrows are data flow for on-demand broadcasting.

live broadcasting, the clients will fetch slides from the storage server when signaled by the embedded scripts.

To prepare for on-demand viewing, a copy of the live stream is sent from the streaming server to the storage server during the capturing process [P2]. The production console also duplicates the script information to a database server. This design allows remote clients to watch the presentation at their own pace during on-demand viewing, because the scripts in the database server can be accessed in any order [P6]. In contrast, the scripts in the AV stream are sequentially embedded and un-indexed, which has limited random accessibility[13].

4.2 Live and on-demand broadcasting

Figure 3 shows a diagram for both live and on-demand broadcasting. The solid arrows depict data flow for live broadcasting, and the dash arrows depict data flow for on-demand broadcasting. In both cases, the web client only has direct access to the streaming server and the web server.

During live broadcasting, the streaming server provides the live stream, and the web server provides the browsing interface, lecture information, and slide images. Lecture information includes speaker biography and the abstract of the talk, which is entered into the database server before the lecture. The slide images, on the other hand, are captured during the lecture (Figure 2) and need to be fetched from the storage server. The on-demand broadcasting data flow is slightly different from live broadcasting. First, the AV steam originates from the storage server rather than the AV encoder. Second, script commands are retrieved from the database server.

During an on-demand session, the user is allowed to view the lecture at his/her own pace. At the very beginning of the session, an XML document is downloaded from the database server to the browser, containing information about all the slide URLs and their time stamps. When the user drags the seek bar of the Window Media player in Figure 1(b), in addition to getting the updated AV stream from the streaming server, a local script computes the corresponding slide from the preloaded XML document and sends a URL request to the webserver to retrieve the image.

[13]As mentioned in Section 6, the live streams are encoded with Window Media Encoder 9. The encoded script commands do not form a separate stream as in [Herlocker and Konstan 1995]. Random accessibility is thus limited because one cannot pre-download all the scripts for slide browsing.

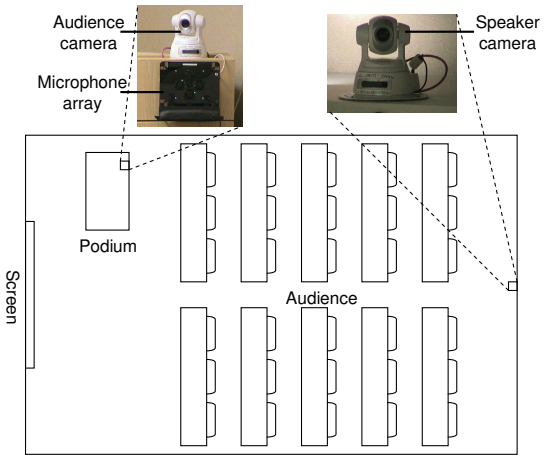


Fig. 4. A typical configuration of the *iCam2* system.

Similarly, if the user wants to synchronize the video with the slide (by pressing the "Sync V" button in Figure 1(b)), a script gets the time of the slide and sends a request to the streaming server for an update.

5. AUTOMATED LECTURE CAPTURING

This section describes the *iCam2* automated lecture content capturing system [P4], which is the core of MSRLCS. In a typical lecture room shown in Figure 4, a speaker camera is placed at the back of the room to capture the speaker, and a camera and microphone array are placed on the podium to capture the audience. Both cameras are Sony SNC-RZ30 PTZ network cameras. These cameras, which can be connected to any networked computer, are capable of sending high resolution (640×480) motion JPEG image sequences at 30 frames per second. The microphone array is an 8-element array. It serves two purposes: 1) capturing audience questions and 2) guiding the audience camera to point to the person asking the question. As mentioned before, the *iCam2* system focuses on the portability issues while improving overall system performance.

In the following, we will present in detail how the audio, visual aids, and video are captured in *iCam2*.

5.1 Audio

Lecture audio in the *iCam* system was captured with two microphones. A wireless microphone captures the speaker [P1]. The second microphone was on the podium and pointed to the audience to capture questions. The two signals, as well as the audio signal from the speaker's laptop, were mixed through an analog audio mixer. A microphone array was installed on the podium, but its only purpose was to guide the audience camera. The system worked, but it did not capture audience questions very well. In particular, it had difficulty capturing questions raised by people in the back of the room, because the microphone on the podium had a limited range.

To address this problem, *iCam2* uses a custom-built microphone array to capture

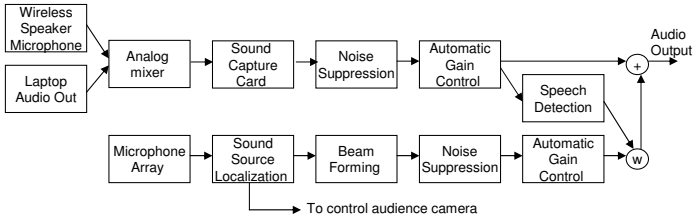


Fig. 5. The audio capturing process.

audience questions. The diagram of the audio capturing process is shown in Figure 5. The speaker sound and the laptop audio output are combined by an analog mixer and captured by a sound card. The captured sound then passes through a noise suppression filter and an automatic gain control algorithm, both implemented in software. The microphone array has 8 microphones. The locations of the microphones are accurately measured. By using the fact that different microphones receive different phases in the signal from the same source position, we can identify the direction of the sound source [Rui and Florencio 2004]. Given that direction, the input signal from the 8 microphones are fused to form a beam pointing to the same direction, which gives much better sound quality [Tashev and Malvar 2005].

The processed audience sound is then digitally mixed with the combined speaker and laptop signal. Unfortunately, adding the two signals directly can cause problems: when the speaker talks or the laptop plays audio, not only the sound card captures the signal, the microphone array also captures the room reverberation, which makes the mixed audio noisy. We therefore designed a weighted mixing scheme to mute the signal from the microphone array whenever a speech or audio signal from the sound card is detected, as illustrated in Figure 5. The multiplication factor W in Figure 5 increases to 1.0 when no speech is detected at the sound card, and reduces to 0.0 otherwise.

To implement the above idea, we adopted an adaptive energy-based speech/noise detection algorithm, which provides for each audio frame (20 ms length) received at the sound capture card a binary decision of speech/noise: $b_1, b_2, \cdots, b_n, \cdots$, where n is the frame index, $b_n = 1$ indicates speech, and $b_n = 0$ otherwise. The weight W at frame n is determined, in a simple form, as:

$$W_n = \begin{cases} W_{n-1} - \alpha, & \text{if } b_n = 1 \\ W_{n-1} + \beta, & \text{if } b_n = 0 \end{cases}$$

where $\alpha = 0.1$ and $\beta = 0.005$ are two empirical parameters. Note the microphone array signal will be completely muted if speech persists for 200 ms. This action is important to reduce the echo. On the other hand, the rate to increase W should be slow, because otherwise the mixing can oscillate for speakers who pause between sentences. Such oscillation can cause unpleasant artifacts.

In general the above algorithm has worked very well. In rare situations, one may still notice artifacts. For example, if the presenter gives short feedback during a question, the audience questioner's volume may vary. In the future we could build a single audio device that contains both the microphone array and the wireless microphone. As long as the signals are synchronized, we could use better solutions

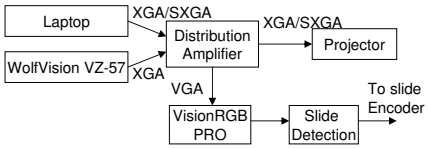


Fig. 6. Flow chart about the visual aids capturing. Please refer to Figure 2 for the slide encoder.

based on techniques such as automatic echo cancellation.

5.2 Visual aids

Slides are captured on-the-fly using an RGB capture device, shown in Figure 6. The design is modular, such that it can capture both slides and material scanned by a WolfVision VZ-57 Visualizer. A distribution amplifier selects output from the visualizer or the laptop and sends it to the projector.

Meanwhile, a copy of the display is digitized by a Datapath VisionRGB Pro capture card [P3]. Currently, images are grabbed at 1 fps. A slide change detection algorithm is run over two subsequent frames. If the difference between two frames is greater than a threshold, we consider it a new slide, and send the production console a message to store this slide as well as informing the AV Encoder to embed a slide change command in the live stream. Note that the slide detection algorithm does not have to be 100% accurate. A slightly over sensitive detection algorithm will do the job well as it will not affect the viewing experience. For the same reason, even if the speaker goes back and forth in the slides, or adds digital marks on the slides during the lecture, the stored images faithfully record what appears in order on the display, which is sufficient for archiving the lecture.

At 1 fps, animations and demos on the screen will not be successfully captured by the video capture card. Currently we capture dynamic contents by pointing the speaker camera at the projected presentation, as explained in Section 5.3.1. This solution works, but can be improved considering that the video capture card is capable of grabbing images at 30 fps. A better solution would be to switch the captured RGB signal at high frame rate into the video codec directly when dynamic contents are shown. We are planning to implement this solution in the near future.

Other types of visual aids, such as writing on a whiteboard, are seldom used by our external speakers. If that happens, the whiteboard contents is captured with the speaker tracking camera that will be discussed in the next section. In the future we may integrate whiteboard capturing techniques such as [He and Zhang 2004; Onishi and Fukunaga 2004; Heck et al. 2006] to our system.

5.3 Video

Video makes lecture viewing more engaging. In *iCam2*, the analog cameras used in the previous *iCam* system are replaced with two network cameras — a speaker camera and an audience camera [P5]. A diagram of the video capturing process is shown in Figure 7. The speaker camera can produce a wide angle view of the lecture room, a close-up view of the speaker, and a view of the projection screen. The audience camera can produce a wide angle view of the audience or a close-up view of an audience member who asks a question. Each camera feeds its video to a virtual

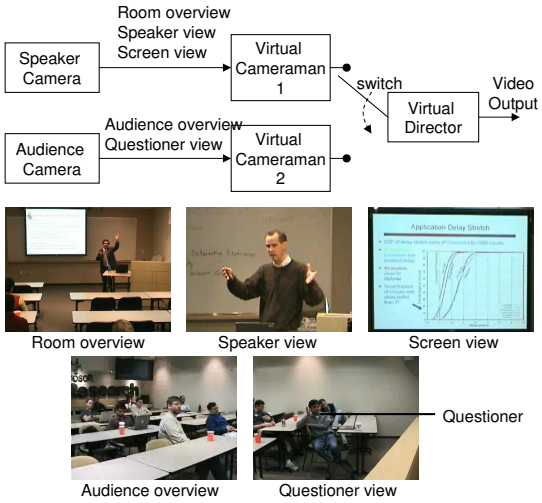


Fig. 7. The video capturing process.

cameraman (VC). These VCs send compressed video to a central virtual director (VD), which selects one as the output video. The video processing components, including the VCs and the VD, are wrapped in a Microsoft Direct Show video source filter, which can be used as a regular video device.

A hybrid tracking algorithm was developed to track the speaker with a single PTZ camera. The cinematography rules are now editable using a scripting language, which makes it possible to port our system to various room configurations.

The remainder of this section describes speaker and audience video capture and the scripting language used by the virtual director.

5.3.1 *Speaker capture.* In the previous *iCam* system, two cameras are used to track the speaker. A static camera was used to track the movement of the speaker. It has a wide horizontal field of view (74 degrees) and covered the front of the lecture room. The other camera was a PTZ camera. Tracking results generated from the first camera guide the movement of the second camera and keep the speaker at the center of the view. Although this approach worked well, dual cameras not only increase the cost and hardware complexity, but they also require manual calibration during setup. In the new system, a single PTZ camera is used to capture both wide angle and close-up views of the speaker. However, there is a research challenge here. To give a high resolution view of the speaker, the PTZ camera can only cover a portion of the front of the room, making it difficult to recover from tracking errors.

We address the single camera speaker tracking problem with a digital/mechanical hybrid scheme. The network camera is operated at a resolution of 640×480. A 320×240 subregion is cropped as the output video based on the location of the speaker. As the speaker moves around within the field of view, the digital cropping region follows in a smooth fashion, thus the name digital tracking. This approach is similar to Yokoi and Fujiyoshi's work [Yokoi and Fujiyoshi 2004]. However, we can also track the speaker mechanically, because the camera is a PTZ camera. Digital

tracking has the advantage of being smooth. Mechanical tracking has the advantage that it can cover a wide area even when the camera is zoomed in to produce a high resolution shot for the speaker. Hybrid tracking achieves the benefit of both worlds.

To improve the aesthetics of the lecture scene, we developed an intelligent pan/zoom selection scheme based on the activity or movement of the speaker. The zoom level of the camera is controlled by the amount of motion detected throughout the tracking. If a speaker stands still most of the time, the camera will zoom in to a close-up shot. On the other hand, if the speaker moves around, it is better to maintain a low zoom level to reduce virtual camera panning. The same speaker tracking camera is used as a screen capture camera, which shows the projected screen when the speaker walks into it, or when the speaker shows an animation or demo. In the former case, the screen shots can show the speaker's gesture to the slides. In the latter case, the video can complement the low frame rate slide image as described in Section 5.2.

Overall, the quality of speaker tracking in *iCam2* is similar to that of the previous *iCam* system, although only a single camera is used. Automatic zoom level control and screen capture are new in *iCam2*. The screen capture feature received much applause from users. For more details about speaker and screen capturing using hybrid tracking, please refer to [Zhang et al. 2005].

5.3.2 *Audience capture.* For a lecture capturing system, it is important to give shots of the local audience from time to time, because they are also part of the event. According to the cinematography rules developed for *iCam*, if the speaker has been on air for a while, the system should switch to the audience view with a certain probability. Five audience overview shot modes are currently implemented in the system: panning from left to right, panning from right to left, and three static views pointing toward the left, center and right side of the room. They are chosen randomly when an audience view is selected by the VD.

When there is a question raised from the audience, professional videographers switch to a shot of the person who asks the question. This switching is accomplished by a sound source localization (SSL) algorithm using the microphone array. In the previous *iCam* system, there are two microphones in the array. In *iCam2*, the number of microphones was increased to eight. Multi-microphone SSL was used to achieve more robust results [Rui and Florencio 2004].

5.3.3 *Scripting for cinematography rules.* The virtual director selects inputs from multiple virtual cameramen according to cinematography rules. In the previous *iCam* system, a five-state finite state machine (FSM) was used to model the transition between cameras. All rules were predefined and hard coded into the system. In *iCam2*, we model the camera transitions using a *nested FSM*. A simple scripting language was developed to describe the rules for switching between different cameras and videos. These improvements greatly enhance the system's portability to room sizes, number of cameras, and camera locations [P5].

Figure 8 shows the nested FSM that models the transition between cameras. As mentioned above, each camera is responsible for a number of tasks, so there are a number of states for each individual camera. For instance, the speaker camera can shoot a room overview, a zoomed speaker view or a screen view; the audience

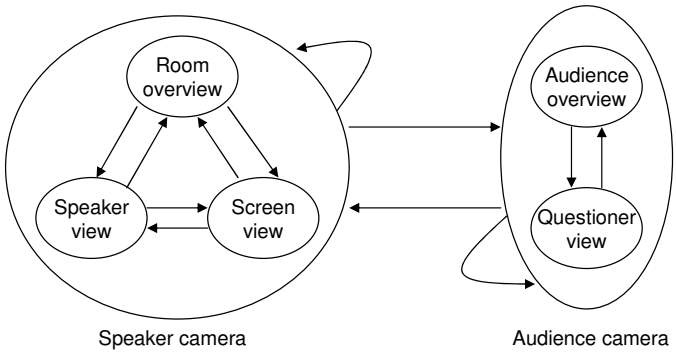


Fig. 8. The nested FSM for modeling the transition between cameras in *iCam2*.

camera can shoot an audience overview or a questioner view. Each camera is thus an FSM, and the transition between different states are driven by events or time. At a higher level, the virtual director controls which camera is on air. There are two states: speaker camera on air or audience camera on air. The transition between these two states are also event and time driven. Note the inner states for each camera and the outer states for the virtual director are not separated. For instance, if the speaker camera plans to zoom into the speaker, it will wait for the outer state to switch to the audience camera to hide the zooming action.

The nested FSM is encoded with a scripting language demonstrated in Figure 9. The description is divided into a few sections:

CAMERAS specifies the various types of cameras used in the capturing process. The current *iCam2* implementation supports two main classes of cameras – `CSpkCamera` and `CAudCamera`, designed to capture the speaker and the audience, respectively. One can use multiple instances of each camera type. The example in Figure 9 uses two cameras one of each type (e.g., SpeakerCam and AudienceCam).

STATES describes the states of the nested FSM (i.e., they represent which camera is on air). The internal states in Figure 8 are defined inside `CSpkCamera` and `CAudCamera`.

EVENTS defines events for each state. For example, when the confidence of SSL `ssl_conf` (defined and computed internally in `CAudCamera`) is below 5, the event `AudLow` is triggered for the audience camera. When the speaker is well tracked, confidence of tracking `spktrack_conf` is above 2 (again computed internally in `CSpkCamera`), the event `SpkHigh` is triggered. The system administrator can adjust these thresholds for different camera locations and room configurations.

TRANSITIONS describes the transitions of states under certain conditions. As shown in Figure 9, the **`when`** statement indicates the current state of the virtual director. The system executes the code in the body of the **`when`** statement when it is in that state.The **`if`** statement specifies the condition under which the rule applies. And, the **`goto`** statement describes a state change. Once every second, the system reviews these rules from top to bottom. If a certain rule is met, a transition is made with the specified probabilities. Consider the first rule as an example. It says that the current state is audience camera on air, and if the speaker camera state

```
# Section describing the used cameras
CAMERAS
CSpkCamera SpeakerCam      # speaker camera
CAudCamera AudienceCam     # audience camera

# Section describing the states
STATES
Audience AudienceCam       # audience camera on air
Speaker SpeakerCam         # speaker camera on air

# Section describing the state events
EVENTS
Audience {
  AudLow: ssl_conf<5,      # audience is silent (SSL confidence is low)
  AudHigh: ssl_conf>5      # audience is talking (SSL confidence is high)
}
Speaker {
  SpkLow: spktrack_conf<2, # speaker is lost (tracking confidence is low)
  SpkHigh: spktrack_conf>2 # speaker is tracked (tracking confidence is high)
}

# Section describing the state transitions
TRANSITIONS
when (Audience) {          # current state is audience camera on air
  if (SpeakerCam.State=SpeakerView &&   # speaker camera is at a speaker view
      AudienceCam.Event=AudLow &&       # audience is silent
      Time>5)                           # audience camera has been on air for
                                        # more than 5 seconds
    goto (Speaker:1.0)                  # change to speaker camera on air with
                                        # probability 1.0
}
when (Speaker) {           # current state is speaker camera on air
  if (Time>90)                          # speaker camera has been on air for
                                        # more than 90 seconds
    goto (speaker:0.3, Audience:0.7)    # randomly switch to audience camera
                                        # on air with probability 0.7, or stay
                                        # with probability 0.3
}
...

INITIALSTATE Speaker       # the initial state is speaker camera on air
MINSHOT 3                  # minimum shot length is 3 seconds
MAXSHOT 1000               # maximum shot length is 1000 seconds
```

Fig. 9. Two example rules encoded with the scripting language.

is showing a speaker view, and the audience camera does not find a questioner, and it has been more than 5 seconds since the audience camera was on air, the virtual director should switch to speaker camera on air with probability 1.0. The identifiers `SpeakerView` and `AudLow` are defined either in previous sections or inside the camera implementation. The second rule in Figure 9 shows the possibility to specify multiple targets for the **`goto`** statement. It says when the current state is speaker camera on air, if the time has passed 90 seconds, the virtual director will switch to speaker camera on air with probability 0.3 and audience camera on air with probability 0.7. At the microscopic level, the camera transition is random, resulting in less predictability, which can make viewing more interesting. At the macroscopic level, some transitions are more likely to happen than others, following the video editing rules. It has been shown that such a strategy performs well in simulating a human director [Rui et al. 2004].

INITIALSTATE defines the initial state when the system starts.
MINSHOT defines the minimum shot duration for the virtual director.
MAXSHOT defines the maximum shot duration.

The scripting language described above is an extension of the work in [Wallick et al. 2004]. It has similar goals as the ASL programming language used in [Machnicki 2002]. That is, to provide the user with a relatively easy method to modify the behavior of the virtual director. The major difference of our work and [Machnicki 2002] lies in the virtual director itself. Future work will investigate the effectiveness of different languages in more detail.

6. SYSTEM IMPLEMENTATION

This section presents more details about the implementation of the MSRLCS system. Four computers are used to run the current system, although they can be merged or duplicated for different scales of service. There is one control computer for management and slide capture, one computer for the streaming server, one computer for the web and database server, and one computer for the archive storage server.

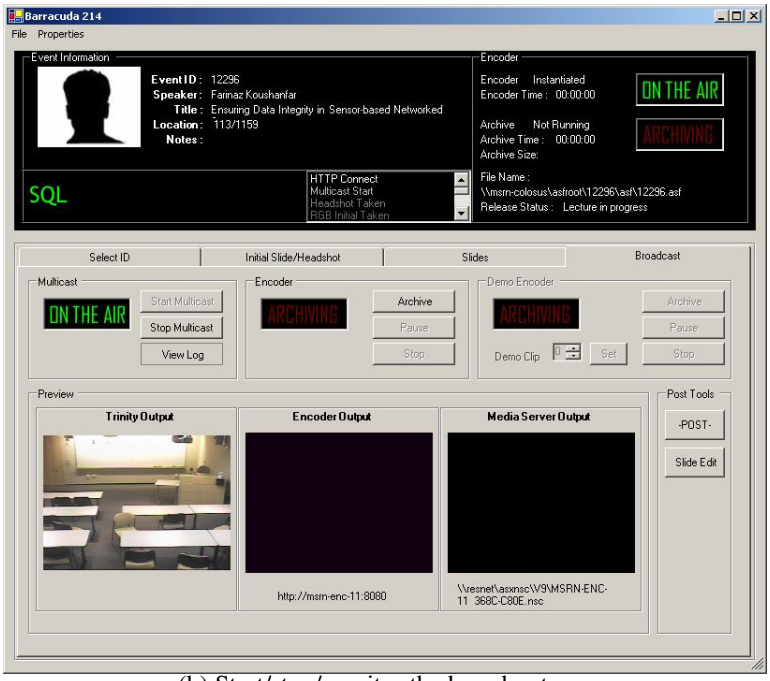
The production console shown in Figure 2 is an application on the control machine that helps manage the whole system. It can be used to select, start, stop a lecture capture, archive a lecture, or monitor the broadcast process, as shown in Figure 10. The control machine has direct connection to the slide capture device. The slide encoder is a simple JPEG compressor, implemented inside the production console.

The streaming server runs on Microsoft Windows Server 2003/Enterprise Edition. It is a Windows Media server that provides multicast capability. The live broadcast is done through multicast, and the on-demand sessions are sent using unicast for interactive user control. For content capture, the *iCam2* automated AV device shown in Figure 2 is programmed as two Direct Show source filters (i.e., audio and video). Inside these source filters sound source localization, beamforming, hybrid tracking, and the virtual director are implemented. From the AV encoder's point of view, the source filters are standard audio and video capture devices. This modular design greatly enhances the flexibility when integrating them into the broadcasting system [P5]. Audio/video encoding uses Microsoft Windows Media Encoder 9 Series. Script command information from the production console is embedded into the stream during the live scenario in order to efficiently push the slide change events to the user. The encoder is instantiated and controlled from the production console using the Distributed Component Object Model. This design allows us to compartmentalize the encoding process and use the best hardware configuration for encoding and steaming the lecture audio and video. Several machines, encoders, and profiles can be instantiated depending on what is required. This implementation allows live streaming for multiple bandwidth targets.

Microsoft Internet Information Services is the web server. The web application serves up HTML, XML and images to Internet Explorer (IE) on the viewer's machine. Some processing is offloaded to IE, which integrates the HTML and XML to render the appropriate page.

The database server runs Microsoft SQL Server (on the same machine as the web server). When a user joins a lecture already in progress, the SQL Server provides

(a) Monitor slide change detection.



(b) Start/stop/monitor the broadcast process.

Fig. 10. Management console for the automated lecture capturing system.

the last captured slide as the first slide rendered when the user joins. There is no waiting for the next slide in order to start. Subsequent slide changes and other script commands are events raised by the Windows Media Player.

In the on-demand scenario, although the script commands exist in the stream, they are ignored by the player. Instead, time driven events such as slide changes are driven by a separate client side component that has access to the Media Player's current time index, the script command, and the slide display user interface (see Figure 1(b)). The component receives relevant events raised by the Windows Media Player that indicate a possible change of state. The component also receives events from the slide display user interface, that can direct the player to seek to a new position based on a selected slide [P6].

The storage server is a computer with a RAID disk array that has 3TB of storage capacity. Each recorded lecture occupies 100-200 MB. The system has recorded over 500 lectures, which used less than 100 GB space.

Since the presentation is archived as it is broadcast, the contents are available for on-demand viewing shortly after the lecture is completed. It typically takes less than a minute, which is mainly due to encoding latency and the 15 to 45 seconds required by the encoder to build indexes in the video/audio stream [P2].

The system also supports streamlined off-line CD image production. The resulting CD has the same look and feel as the online client without the network dependency. These CDs are popular with the speakers themselves and are often a motivating factor in their agreement to be recorded for wider distribution.

7. SYSTEM USAGE

The MSRLCS system was first deployed on June 14, 2001. It has captured 522 lectures by mid-August, 2005, which is about 10 lectures per month. A total of 20,383 sessions have been viewed online, among which 11,115 (54.5%) are live sessions, and 9,268 (45.5%) are on-demand sessions. This breakdown is in contrast to the results reported in the BIBS system at UC Berkeley, where live plays contributed roughly 10%. BIBS was not designed to replace attendance at live lectures, and the students primarily use the system for on-demand replay when studying for exams.

Figure 11 shows the number of lectures broadcast per quarter, the average number of live sessions per lecture per quarter, and the average number of on-demand sessions per lecture per quarter. It can be seen that the number of live sessions is relatively stable – between 20 and 30 per lecture. The talks are typically specialized research talks and not targeted to the general public, which explains the relatively low usage. However, the number is significant considering the local audience in the lecture room is often less than 20[14].

Figure 12 shows the distribution of lectures with respect to the number of viewing sessions including both live and on-demand sessions. For instance, there are 55 lectures that have only $0 \sim 5$ viewing sessions, while there are 28 lectures that have more than 100 viewing sessions. These statistics suggest which lectures are worth keeping.

[14]Interestingly, the Berkeley MIG Seminar webcast between 1996-2002 observed similar results — the size of the audience in the lecture room is about the same size as watching remotely [Rowe 2006].

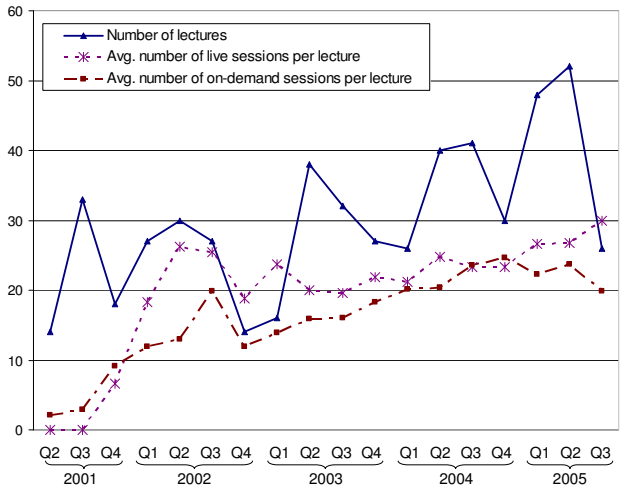


Fig. 11. Lecture and viewing statistics.

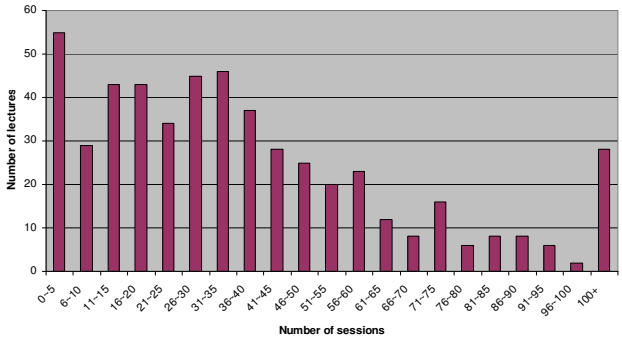


Fig. 12. Distribution of lectures w.r.t. the number of viewing sessions.

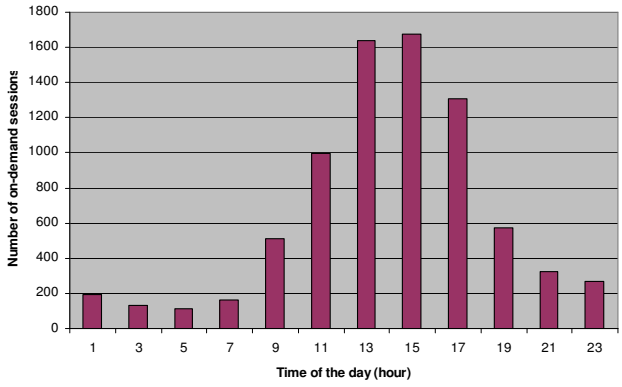


Fig. 13. Time-of-the-day distribution for on-demand sessions.

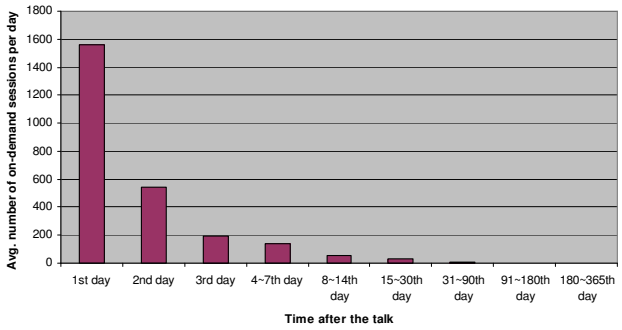


Fig. 14. When do on-demand sessions happen after the lecture.

Figure 13 shows the time-of-the-day distribution for people watching on-demand lectures. It appears that most people like to watch them in the afternoon.

Figure 14 shows how soon people watch an on-demand lecture after the actual presentation. The statistics are collected on the 345 lectures captured before August 2004. The vertical axis is the average number of on-demand sessions per day in a certain time period. The trend is obvious: lectures are watched less often when time passes. Around 54.8% of the sessions happen within 1 week after the talk, 72.3% of them are within 1 month, and 80.2% are within 3 months. On average, less than 1.6 viewing sessions were made per day for talks older than six months. After one year, 152 out of the 345 lectures (44.1%) are never visited again. These findings are consistent with the work in [Cherkasova and Gupta 2002], which found a similar skewed patten of access.

8. CONCLUSIONS

We have described an automated end-to-end lecture capturing and broadcasting system. The contribution of this work is two-fold. First, we proposed a system architecture that minimizes pre- and post-production time. It allows remote clients to view lectures both live and on-demand. Second, we reported on recent progress on automating the lecture capture process, namely the *iCam2* system. Great effort has been made to enhance the system's portability while improving overall performance. The result is a system that automated both lecture capture and broadcasting, thereby reducing the cost of operation.

There are still many interesting topics that remain to be explored. For instance, dynamic presentation materials could be better captured if we incorporate the RGB signal into the video stream directly. Detecting the speaker's head orientation and gesture reliably can improve speaker tracking and video switching. The MSRLCS system has been set up in one of our lecture rooms. Another area of future work is to deploy it in several rooms with different audio/video configurations. We are also working on the combination of *iCam2* and the ConferenceXP platform[15], which will allow remote users to have their own preference on the cinematography rules and will allow two-way interaction with the speaker.

[15]http://www.conferencexp.net/

A complete end-to-end lecture capturing and broadcasting systems will make a big impact on how people attend and learn from lectures. We envision that capturing and broadcasting technologies will continue to advance, and making a presentation available online will be as easy as turning on a light switch.

9. ACKNOWLEDGMENTS

Many thanks to anonymous editors and reviewers for insightful suggestions on improving the paper quality.

REFERENCES

Abowd, G. 1999. Classroom 2000: an experiment with the instrumentation of a living educational environment. *IBM Systems Journal 38,* 4, 508–530.

Amir, A., Ashour, G., and Srinivasan, S. 2004. Automatic generation of conference video proceedings. *Journal of Visual Communication and Image Representation 15*, 467–488.

Baecker, R. 2003. A principled design for scalable internet visual communications with rich media, interactivity and structured archives. *Proc. Centre for Advanced Studies on Collaborative Research*.

Bianchi, M. 1998. Autoauditorium: a fully automatic, multi-camera system to televise auditorium presentations. *Proc. Joint DARPA/NIST Smart Spaces Technology Workshop*.

Bianchi, M. 2004. Automatic video production of lectures using an intelligent and aware environment. *Proceedings of the 3rd international Conference on Mobile and Ubiquitous Multimedia*, 117–123.

Cherkasova, L. and Gupta, M. 2002. Characterizing locality, evolution and life span of accesses in enterprise media server workloads. *Proc. of NOSSDAV*.

Cruz, G. and Hill, R. 1994. Capturing and playing multimedia events with streams. *Proc. ACM Multimedia*, 193–200.

Finn, K., Sellen, A. J., and Wilbur, S. 1977. *Video-Mediated Communication.* Lawrence Erlbaum.

Gleicher, M. and Masanz, J. 2000. Towards virtual videography. *Proc. ACM Multimedia*, 375–378.

He, L., Grudin, J., and Gupta, A. 2001. Designing presentations for on-demand viewing. *ACM Conference on Computer Supported Cooperative Work (CSCW)*.

He, L.-W. and Zhang, Z. 2004. Real-time whiteboard capture and processing using a video camera for teleconferencing. *Microsoft Research Technical report, MSR-TR-2004-91*.

Heck, R., Wallick, M., and Gleicher, M. 2006. Virtual videography. *ACM Trans. on Multimedia Computing Communications and Applications*, to appear.

Herlocker, J. L. and Konstan, J. A. 1995. Commands as media: Design and implementation of a command stream. *Proc. ACM Multimedia*, 155–165.

Liu, T. and Kender, J. R. 2004. Lecture videos for e-learning: Current research and challenges. *Proceedings of IEEE International Workshop on Multimedia Content-based Analysis and Retrieval*.

Machnicki, E. 2002. Virtual director: automating a webcast. *Proc. SPIE Multimedia Computing and Networking*.

Matsuo, Y., Amano, M., and Uehara, K. 2002. Mining video editing rules in video streams. *Proc. ACM Multimedia*, 255–258.

Mukhopadhyay, S. and Smith, B. 1999. Passive capture and structuring of lectures. *Proc. ACM Multimedia*, 477–487.

Onishi, M. and Fukunaga, K. 2004. Shooting the lecture scene using computer controlled cameras based on situation understanding and evaluation of video images. *International Conference on Pattern Recognition (ICPR)*.

Rowe, L. A. 2006. Personal communication with the authors.

Rowe, L. A. and Casalaina, V. 2006. Capturing conference presentations. *IEEE Multimedia 13,* 4.

Rowe, L. A., Pletcher, P., Harley, D., and Lawrence, S. 2001. BIBS: a lecture webcasting system. *Technical report, Berkeley Multimedia Research Center, U.C. Berkeley*.

Rui, Y. and Florencio, D. 2004. Time delay estimation in the presence of correlated noise and reverberation. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Rui, Y., Gupta, A., Grudin, J., and He, L. 2004. Automating lecture capture and broadcast: technology and videography. *ACM Multimedia Systems Journal 10,* 1, 3–15.

Rui, Y., He, L., Gupta, A., and Liu, Q. 2001. Building an intelligent camera management system. *Proc. ACM Multimedia*, 2–11.

Scott, P. and Mason, R. 2001. Graduating live and on-line: the multimedia webcast of the open university's worldwide virtual degree ceremony. *Proc. Webnet – World Conference on the WWW and Internet*.

Steinmetz, A. and Kienzle, M. 2001. The e-seminar lecture recording and distribution system. *Proc. SPIE Multimedia Computing and Networking 4312*.

Tang, J. and Issacs, E. 1993. Why do users like video? Studies of multimedia-supported collaboration. *Computer Supported Cooperative Work: An International Journal 1,* 3, 163–196.

Tashev, I. and Malvar, H. 2005. A new beamforming design algorithm for microphone arrays. *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Wallick, M., Rui, Y., and He, L. 2004. A portable solution for automatic lecture room camera management. *Proc. ICME*.

Wang, F., Ngo, C. W., and Pong, T. C. 2003. Synchronization of lecture videos and electronic slides by video text analysis. *Proc. ACM Multimedia*, 315–318.

Yokoi, T. and Fujiyoshi, H. 2004. Virtual camerawork for generating lecture video from high resolution images. *Proc. ICME*.

Zhang, C., Rui, Y., He, L., and Wallick, M. 2005. Hybrid speaker tracking in an automated lecture room. *Proc. ICME*.

Manuscript received in Dec. 2005.