AN INTERACTIVE REGION-OF-INTEREST VIDEO STREAMING SYSTEM FOR ONLINE LECTURE VIEWING

Aditya Mavlankar, Piyush Agrawal, Derek Pang, Sherif Halawa, Ngai-Man Cheung and Bernd Girod

Information Systems Laboratory, Department of Electrical Engineering
Stanford University, Stanford, CA 94305, USA
Email: aditya.mavlankar@ieee.org, {piyushag, dcypang, halawa, ncheung, bgirod}@stanford.edu

ABSTRACT

ClassX is an interactive online lecture viewing system developed at Stanford University. Unlike existing solutions that restrict the user to watch only a pre-defined view, *ClassX* allows interactive pan/tilt/zoom while watching the video. The interactive video streaming paradigm avoids sending the entire field-of-view in the recorded high resolution, thus reducing the required data rate. To alleviate the navigation burden on the part of the online viewer, *ClassX* offers automatic tracking of the lecturer. *ClassX* also employs slide recognition technology, which allows automatic synchronization of digital presentation slides with those appearing in the lecture video. This paper presents a design overview of the *ClassX* system and the evaluation results of a 3-month pilot deployment at Stanford University. The results demonstrate that our system is a low-cost, efficient and pragmatic solution to interactive online lecture viewing.

Index Terms— IRoI video, lecture capture systems, tracking, object recognition, slide synchronization

1. INTRODUCTION

Growing Internet access, increasing network throughput, improving computer hardware and enhanced video compression are providing a boost to inexpensive online delivery of lecture videos. However, most lecture capture systems depend on human camera operators as well as manual work of post-production and online video publishing, thus resulting in expensive solutions. Some systems employ a static, unmanned camera that records a limited field-of-view, thus confining the lecturer's movement. More recently, it has been proposed to employ a high-spatial-resolution unmanned camera with a wide field-of-view, followed by automatic cropping to simulate a human camera operator [1]. Note that cropping is necessary since the entire field-of-view, recorded in high resolution, entails prohibitive bit-rate for streaming to a remote client, and is also unsuitable for display, unless the client has a large display screen.

This work was done when Aditya Mavlankar was a doctoral candidate at Stanford University.

The *ClassX* lecture capture system grew out of the observation that, the video, generated with or without a human camera operator, might not show the portion of the frame that a particular user wants to watch. *ClassX* solves this problem by allowing each user to independently control pan/tilt/zoom while watching the video. Thus, each user can interactively choose an arbitrary region-of-interest (RoI). Interactive region-of-interest (IRoI) allows the user to take advantage of the wide field-of-view as well as the recorded high resolution, while requiring modest transmission data rate. Also, the display screen at the user's end is not required to be large.

We surveyed lectures on science, mathematics and engineering topics and found that, typically, the lecturer writes on several boards in the classroom. Besides, digital presentation slides might be projected next to the boards. Figure 1 illustrates a typical podium with multiple boards. While watching lectures with the *ClassX* client, each user can focus on an RoI of her choice, thanks to IRoI functionality. For instance, Fig. 1 shows screens of three viewers watching different RoIs.

A straightforward way to serve IRoI video is to decode the high-spatial-resolution video followed by cropping out the RoI sequence according to the client's commands and encoding and transmitting this sequence. However, this simple approach does not scale, since RoI video encoding needs to be invoked individually for each user. In our own work, we have proposed spatial-random-access-enabled video coding, which allows the server to encode the recorded field-of-view once with multiple resolution layers to support different zoom factors [2–4]. However, with the coding scheme in [2–4], the transmitted bit-stream is neither standard compliant nor is it simply comprised of multiple standard compliant video bit-streams. Hence, it is not possible to use decoders available in a web-browser to decode the received RoI video. In this paper, we propose a coding scheme that preserves spatial random access while still allowing RoI video decoding by invoking one or more instances of a standard decoder available in popular web-browsers. The *ClassX* video player has been tested inside Firefox, Safari, IE and Chrome web-browsers and requires a one-time installation of the Mi-

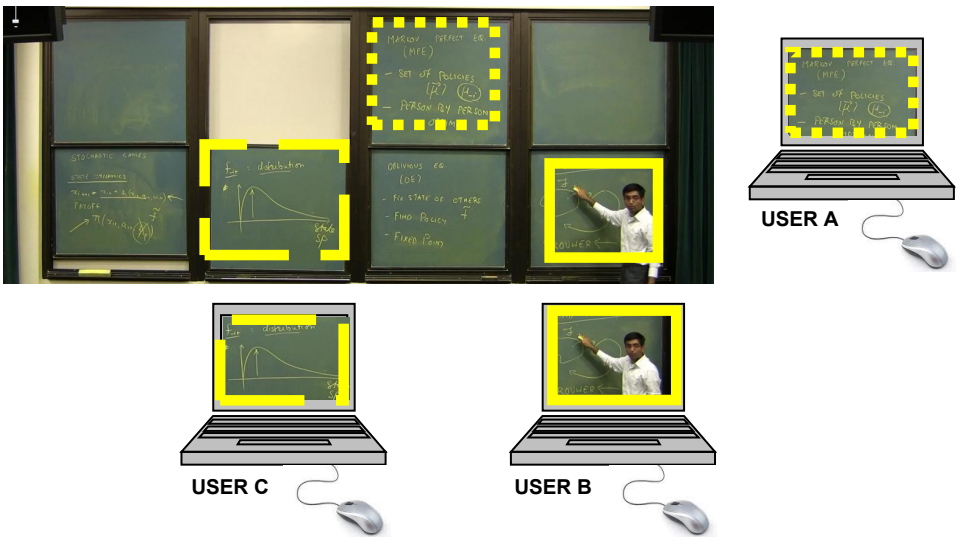


Fig. 1. A typical podium consisting of multiple writing boards. Sample RoIs chosen by three viewers.

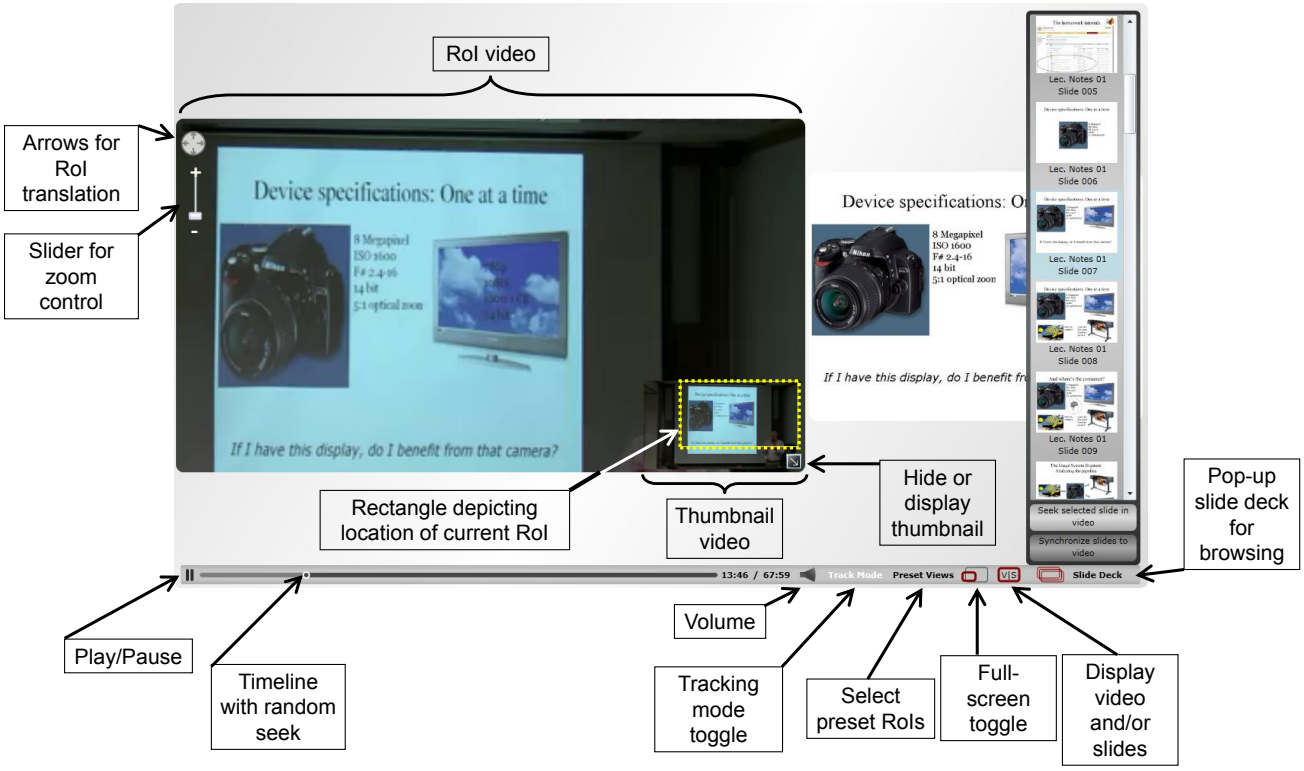


Fig. 2. Screenshot of the *ClassX* video player and available controls.

crosoft Silverlight plug-in[1]. Figure 2 shows a screenshot of the *ClassX* video player and available controls.

[1]Although we chose Silverlight [5] platform for the *ClassX* system, it is feasible to employ Adobe's Flash [6] platform instead. Employing Flash would require modification of the *ClassX* client software. Since the *ClassX* player can be embedded in an HTML web-page, the user is unaware of which platform is used, as long as the plug-in is installed in her web-browser.

Apart from allowing the user to control pan/tilt/zoom, *ClassX* offers a *Tracking* mode. The RoI video streamed in *Tracking* mode is generated through automatic cropping and mimics a human camera operator, similar to the approach in [1]. This approach differs from prior work employing a camera that physically moves to track the lecturer or multiple cameras that cover different regions [7–9].

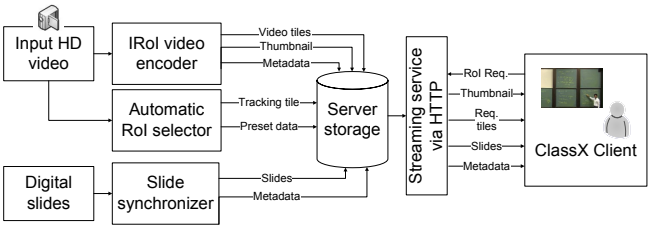


Fig. 3. Overview of system architecture.

ClassX can also ingest a deck of digital presentation slides and automatically synchronize slide display with the video. We employ computer vision techniques for estimating which slide is shown during which time segment in the video. The *ClassX* player can automatically advance slides during video playback. The user can also select a slide and jump to the video segment showing that particular slide.

The paper is organized as follows. Section 2 provides an overview of the system architecture. Section 3 covers the video coding scheme and discusses the aspects that lower encoding time as well as bit-rate. Section 4 explains how *ClassX* generates video associated with the *Tracking* mode. The slide recognition algorithm as well as slide presentation features of the *ClassX* player are discussed in Section 5. We have employed *ClassX* for recording and publishing several lecture courses at Stanford University. We have analyzed usage patterns and extracted several interesting metrics, which are reported in Section 6.

2. OVERVIEW OF SYSTEM ARCHITECTURE

Figure 3 provides an overview of the *ClassX* system. The high-resolution video, recorded using an HD camcorder, is first decoded and fed to the IRoI video encoder. The IRoI video encoder creates multiple resolution layers. Each layer is subdivided into non-overlapping tiles. Each tile is independently compressed using H.264/AVC [10]. A thumbnail overview is also created and encoded using an H.264/AVC encoder. Similarly, a tile of the video in *Tracking* mode is also generated. A client receives the thumbnail video at all times along with tiles that are relevant for rendering the chosen RoI. If the user selects the *Tracking* mode, the client receives the thumbnail and the *Tracking* tile.

At the client's side, multiple instances of an H.264/AVC decoder can be instantiated for decoding received tiles as well as the thumbnail. The rendering of multiple tiles is accomplished by simultaneous playback of multiple videos. The client synchronizes playback and controls the scaling and placement of received tiles in correspondence with the chosen RoI. The tile portions falling outside the RoI are not rendered. The streamed tiles belong to the resolution layer that is determined by the chosen zoom factor. Additional resampling employed at the client allows storing few resolution

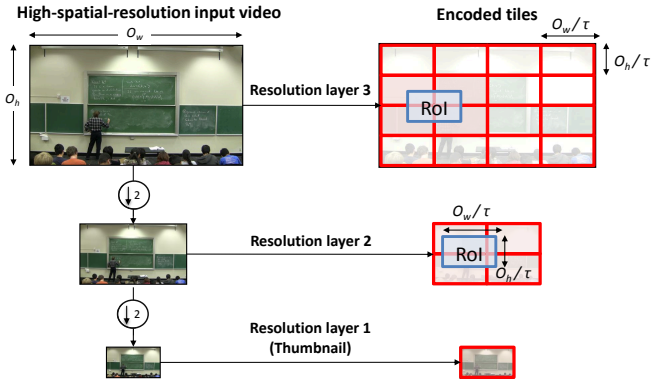


Fig. 4. Proposed coding scheme shown for the case of three resolution layers. The original high-spatial-resolution video is dyadically downsampled to create multiple resolution layers. Each resolution layer is further subdivided into tiles. Each tile is independently coded using an H.264/AVC video encoder.

layers at the server while allowing continuous adjustment of the zoom factor.

For serving relevant tiles to the clients, we employ an Apache HTTP (hypertext transfer protocol) server [11] with a module that serves tiles with specified time segments. Note that a newly needed tile can only be decoded starting from an intracoded frame. In case a given tile is not available at the client, we fill in missing pixels by upsampling relevant parts of the thumbnail. Metadata describing locations of intracoded frames are sent for each tile, so the client can request appropriate time segments for the relevant tiles.

Digital presentation slides are analyzed and matched against the video to determine slide and video synchronization information. Since the slide deck consumes little data rate, it is transmitted in its entirety when the player starts up. Metadata holding synchronization information is also transmitted to the client.

3. IROI VIDEO STREAMING

3.1. Proposed Coding Scheme

The server performs a one-time encoding of the thumbnail and the tiles and relevant streams can be served to different users depending on their individual RoIs. As shown in Fig. 4, the input video of size $o_w \times o_h$ is dyadically downsampled to create K resolution layers in all. The lowest resolution layer, which provides a thumbnail overview, is transmitted at all times to aid navigation within the recorded field-of-view. Let $\tau = 2^{K-1}$. Each tile has spatial dimensions of $\frac{o_w}{\tau} \times \frac{o_h}{\tau}$ pixels, which is also equal to the size of the RoI display portion at the client's side. The tile size, RoI display size, and choice of the resolution layer guarantee that any chosen RoI overlaps at most four tiles from the relevant layer. Hence, includ-

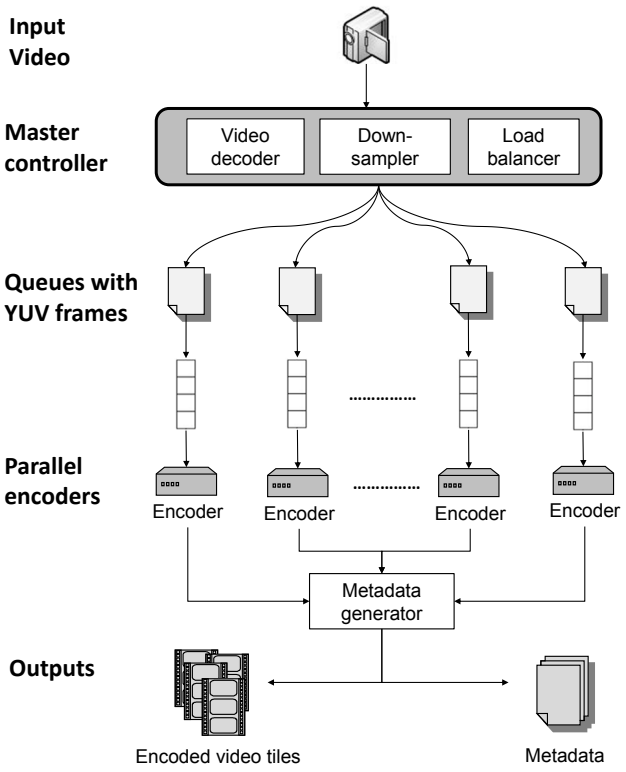


Fig. 5. Proposed architecture of a multi-threaded IRoI video encoder.

ing the thumbnail, the client needs to decode and synchronize at most five H.264/AVC video streams, each having spatial dimensions of $\frac{o_w}{\tau} \times \frac{o_h}{\tau}$ pixels. The client reduces required bit-rate by not fetching a tile if the RoI overlaps less than 5% of the area on the tile. As noted above, in the *Tracking* mode, only two tiles (including the thumbnail) are fetched, with a *Tracking* tile that is larger than the other tiles.

3.2. Parallelization Employed by IRoI Encoder

With multiple tiles and resolution layers, IRoI video encoding entails higher complexity than coding a single HD video. An efficient video encoder is essential for timely online publishing of lecture videos. Since each tile is encoded independently, *ClassX* can parallelize tile encodings. Additionally, it is efficient to crop out all tiles from the camera-recorded video at the same time. Figure 5 illustrates our IRoI encoder design. The encoder follows a multi-threaded software implementation containing a master controller that distributes the task of encoding each tile to a single thread. A thread can be processed by any core within a CPU, enabling efficient parallelization. During the encoding process, the master controller first decodes the input video file into raw YUV frames. Secondly, it downsamples the raw YUV frames into different resolution layers and then subdivides layers into appropriate tiles. Each raw tile is put into a queue that belongs to the

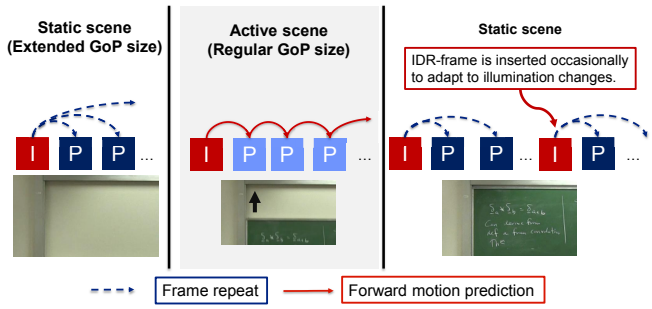


Fig. 6. Proposed coding structure using adaptive frame skipping. The generated bit-stream corresponding to each tile is H.264/AVC compliant as before.

corresponding encoder thread. Each encoder thread encodes the respective tile into an H.264/AVC compliant bit-stream. Lastly, all the encoded tiles are passed to the metadata generator, which builds the appropriate file container and annotates the video with relevant information, such as the video time length and the position of the intracoded frames.

3.3. Adaptive Frame Skipping for IRoI Video Coding

Lecture videos often contain large regions with static backgrounds and small regions with active motion. Hence, it is beneficial to apply conventional motion-compensated predictive coding only for tiles that contain motion. For the static tiles, we can simply encode a single static background frame, and skip over all the subsequent static frames until the tile is active again. The frame skipping approach lowers the encoding complexity as well as the storage requirement of the IRoI video without affecting its perceptual visual quality.

While incorporating adaptive frame skipping, we can still generate an H.264/AVC standard compliant bit-stream for each tile as depicted in Fig. 6. We determine an input video frame to be part of a static scene or an active scene by measuring its motion change compared to the last displayed frame. Motion change can be detected by taking the sum of absolute pixel differences and applying an appropriate threshold in a block-based manner. If all blocks are classified as static then the entire frame is classified as static. The block-based method allows detecting small local variations and also enables early termination of the decision process. When a video segment is active, the video frames are encoded using standard H.264/AVC structure with a regular group of pictures (GoP) size to allow temporal random access. When a segment becomes static, an IDR frame is inserted as a static background frame. The IDR frame serves as a temporal random access point and subsequent P frames are forced to use the P-skip mode for each macroblock, thus avoiding motion estimation and motion compensation. In case of a static scene, the encoder can also further extend the GoP size to eliminate the coding of redundant I frames representing the

same scene. This method is similar to the long-term memory background-based motion compensation in [4] except that the I-frames are refeshed when changes occur. When a static tile becomes active, the encoder reverts to the normal forward prediction and the regular GoP size.

4. AUTOMATIC ROI SELECTOR

For ease of use, *ClassX* offers two modes of automatic RoI selection: 1) *Tracking* mode, which mimics a human camera operator by generating a video that follows the lecturer, and 2) *Preset* mode, which allows users to choose fixed preset RoIs from a set of pre-determined locations.

4.1. Tracking mode

The *Tracking* mode entails streaming a *Tracking* tile containing a "recommended view". The *Tracking* tile is cropped directly from the input video and mostly tracks the lecturer when he is visible in the recorded field-of-view. Although the RoI in the *Tracking* mode is non-stationary, it does not entail switching tiles once the *Tracking* mode is enabled.

To automatically identify our tracking target, a server-side algorithm first detects a set of target points based on prominent features of the lecturer, such as motion and skin color, over a set of frames sampled from the video. The target points are then grouped together by k-means clustering to generate a set of matching templates. After we acquire the matching templates, we find the starting location of the lecturer through template matching at the beginning of the video sequence.

After the starting location of the lecturer is identified, we can initiate lecturer tracking by using dynamic background subtraction. First, we set the lecturer as the foreground of the scene. We then apply background subtraction with running Gaussian average [12] to track the lecturer in subsequent frames. Since the trajectory of the lecturer is continuous and is often constrained to a horizontal direction, we reduce our tracking problem into a 1-D problem and estimate the location of the lecturer by calculating the horizontal centroid of the foreground pixels. Factors, such as occlusion and motion of non-target objects, can cause our tracker to drift from the target. To prevent this problem, we compute the matching confidence between the templates and the estimated target. When the matching confidence is high, we update our template set by removing an older template and adding the current target. This dynamic feedback provides the system with latest information, such as target orientation and illumination changes, and enables improved tracking accuracy without requiring a large set of matching templates. When the matching confidence is low, we try to find the best matching region through template matching. If no match is found, we do not update the location of the recommended view as well as the template and the background model. In addition to generating the viewing trajectory, we determine an appropriate zoom factor for the recommended view based on the size of our detected target and the motion activity surrounding the target.

To reduce computational burden, we process every fifth video frame for lecturer tracking. A post-processing step, involving low-pass filtering and trajectory interpolation, is applied to interpolate and stabilize the viewing trajectory. After the trajectory is finalized, the system crops out the appropriate region from the input video and encodes the sequence into a *Tracking* tile using an H.264/AVC encoder.

4.2. Preset mode

Instead of following the lecturer, the *Preset* mode offers the choice of different RoIs centered around different writing boards or projection screens. Viewers can easily switch between these views and focus on different lecture material printed on different boards. To find the set of appropriate presets, our server-side algorithm applies the Canny filter and determines a set of closed rectangular contours in the video. The coordinates of the selected rectangular regions are then streamed as metadata corresponding to preset views. Since *Preset* mode entails a non-moving RoI, an independent tile is not generated.

5. SLIDE RECOGNITION

To enhance the user experience, *ClassX* supports the display of electronic slides alongside relevant sections of the lecture video. When a slide transition occurs in the lecture video, the slide image is updated accordingly. *ClassX* also allows the user to select a slide and access the time segment of the video where the selected slide is discussed.

To enable slide synchronization, we perform automatic slide recognition offline by matching video frames with electronic slides. Slide recognition involves two steps: 1) extraction of keyframes from the video and 2) matching keyframes to a deck of slides using pairwise image comparison. We first extract keyframes from where slide transitions occur in the video. To detect slide transition events, we compute frame difference between successive frames, and declare a slide transition when the difference exceeds a threshold. To reduce the probability of false positive due to the movement of foreground objects (e.g., instructor's motion), we locate foreground objects by comparing the current frame with a background model and exclude them from computing the frame difference in transition detection.

With the detected transitions and the extracted keyframes, we then match the keyframes against a deck of electronic slides to identify the ones being displayed in the video frames. In particular, we use pairwise image comparison to perform the matching [13]. We extract local descriptors from a keyframe. We detect local extrema in the difference-of-Gaussian (DoG) filtered images and select interest points from these extrema [14]. We compute the 128-dimensional

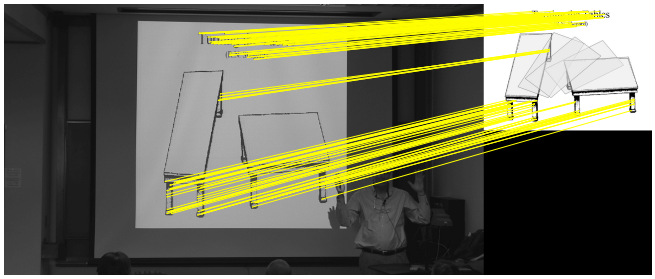


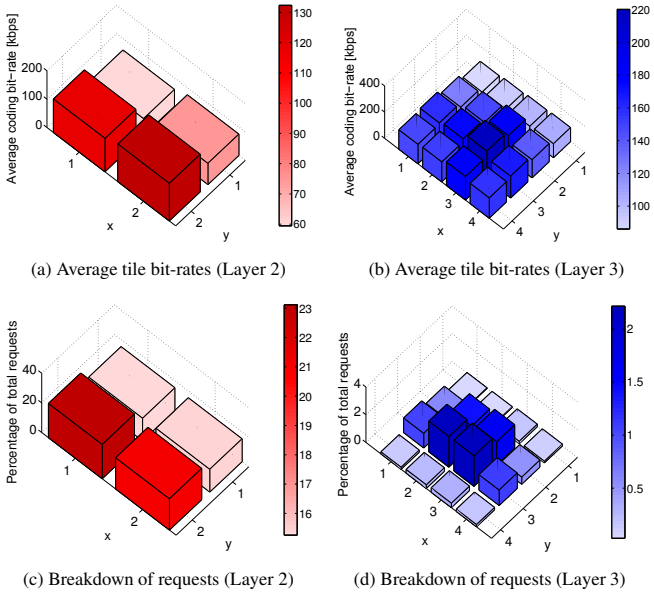
Fig. 7. Matching interest points between a keyframe and an electronic slide.

SIFT descriptor vectors to summarize the local gradients around these interest points. These descriptors are invariant to image scaling and rotation, and are partially invariant to illumination and viewpoint change. Then, we compare the extracted descriptors from the video frame against those from the slides. We establish correspondences between descriptors that are nearest neighbors in the descriptor space (Figure 7). From these correspondences, we use RANSAC to estimate the geometric transformation between the frame and the slides [13]. We reject outlier correspondence pairs that are inconsistent with the estimated transformation. We return as the matching result the slide which has the maximum number of correspondences consistent with estimated transformation.

6. PILOT PROJECT EVALUATION

Since September 2009, we have engaged in pilot deployment of *ClassX* at Stanford University. To assess the techniques presented in this paper, we analyze usage data gathered between January 2010 and April 2010. During this period, we covered two courses offered by the Department of Electrical Engineering. Two distinct lecturing styles were encountered in these two courses. One style mainly relies on digital slide presentation and the other style exclusively uses blackboards. For these two courses, our system published a total of 37 lectures. Each lecture is about 75 minutes long and is recorded by a static camera with resolution of 1920×1080 pixels and at 30 frames/sec. The camera compresses the high-resolution videos using H.264/AVC at 24 Mbps. Our IRoI encoder decodes the recorded bit-stream and creates 3 dyadic resolution layers with a total of 21 tiles. Each tile has 480×270 pixels. The *Tracking* tile is generated by the automatic RoI selector with a resolution of 960×540 pixels. Each tile is encoded into an H.264/AVC bit-stream using FFmpeg and the x264 (v0.77) codec library. The motion estimation is set to have quarter-pixel accuracy with a search range of 16 pixels. The encoded videos have an intraframe period of 59 frames with 16 B frames between the anchor frames.

The IRoI encoder runs on a dedicated computer equipped with an Intel Xeon Quad-Core 5140 2.33 GHz processor with 4 GB of RAM. Using the parallelization technique described



(a) Average tile bit-rates (Layer 2)

(b) Average tile bit-rates (Layer 3)

(c) Breakdown of requests (Layer 2)

(d) Breakdown of requests (Layer 3)

Fig. 8. Average coding bit-rate and percentage of total requests for each tile from resolution layers 2 and 3. The x and y coordinates correspond to the spatial location of each tile relative to the top left corner of the video.

in Section 3.2, our IRoI encoder needs an average encoding time of 50 ms per input frame. The processing times for automatic RoI selection and slide recognition are 10 ms and 8.25 ms per input frame respectively. As a result, the total processing time for a 75-minute lecture is only about 140 minutes.

On average, for a 75-minute lecture, the encoded video with 22 tiles requires 1.95 GB of storage space. The average coding bit-rates per tile are 184, 96, 146 and 570 kbps for Layer 1, Layer 2, Layer 3 and the *Tracking* tile respectively. The average coding rates for each tile in Layers 2 and 3 are shown in Figs. 8 (a) and 8 (b). The average PSNR of all tiles is 37.24 dB compared to the input of the tile encoder.

We also evaluated the effectiveness of adaptive frame skipping for IRoI video. We implemented the method as described in Section 3.3 by modifying the x264 library and compared its performance against x264. The experiment is performed on two 15-minute video segments extracted from each of our published courses. The videos are encoded into altogether 21 tiles across the 3 resolution layers. The encoder settings are identical except for the differences in their coding structures. The P-skip mode is also enabled in both encoders. Figure 9 shows that adaptive frame skipping has an average coding gain of 2 dB at low bit-rates and 0.6 dB at high bit-rates over all encoded tiles. In terms of encoding complexity, Fig. 10 indicates that adaptive frame skipping can reduce the average encoding time spent per frame by up to 40% at low quality and by up to 30% at high quality. These results demonstrate that adaptive frame skipping helps in reducing

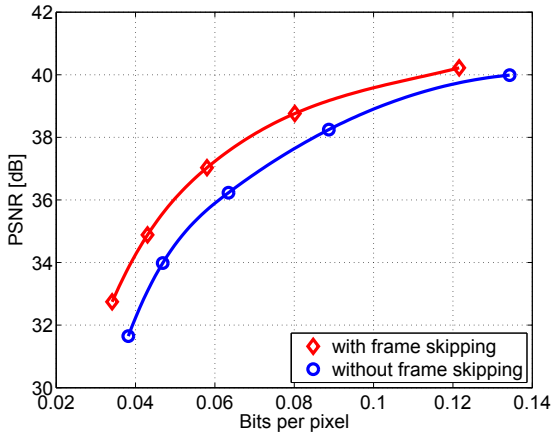


Fig. 9. PSNR with and without adaptive frame skipping.

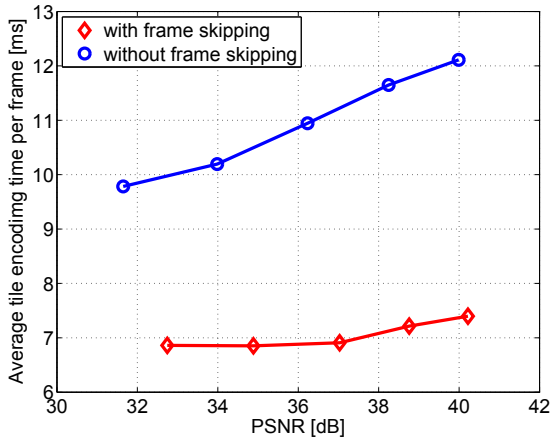
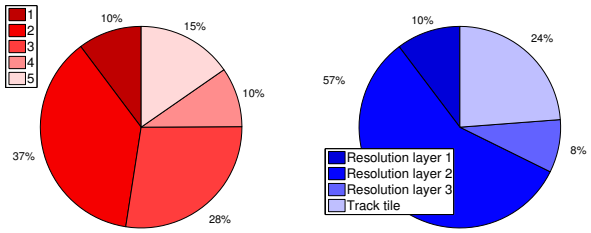


Fig. 10. Encoding time with and without adaptive frame skipping. The encoding time is the average encoding time spent per frame per tile.

both storage space as well as encoding complexity of IRoI videos, while providing H.264 compliant bit-streams.

During the pilot deployment, the *ClassX* streaming server received a total of 84,064 video streaming requests from 782 unique IP addresses. The mean transmission bit-rate per streaming session is 493 kbps with a standard deviation of 246 kbps. The highest observed bit-rate is 2342 kbps and the lowest observed bit-rate is 133 kbps. The optimization of fetching only those tiles having significant overlaps with the RoI also plays a role in curbing required bit-rate. Each user requires about 2.82 tiles on average. The percentage breakdown of the numbers of tiles streamed per request is shown in Fig. 11 (a). We measured the average tile switching delay to be 3.4 seconds with a standard deviation of 1.9 seconds. The tile switching delay is the time between an RoI change and when all newly required tiles start playing back. As described previously, during the switching time, missing tiles are made up for by using error concealment from the thumbnail video.



(a) Number of tiles streamed per request

(b) Breakdown of requests according to layer

Fig. 11. Average number of tiles streamed per request and breakdown of requests according to layer.

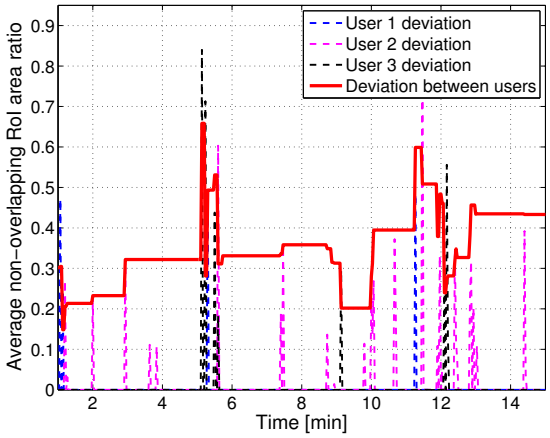


Fig. 12. Deviation measure of RoIs for three individual users and between five users. The vertical axis represents the area ratio of the non-overlapping part of the RoI with a reference RoI. In case of an individual user, the reference RoI is from the previous frame. Given a pair of users, the deviation measure is computed by comparing the RoIs of the two users. The plotted trace labeled "Deviation between users" is obtained by averaging the numbers corresponding to all possible pairs that can be formed with five users.

Figure 11 (b) shows the demand for the different resolution layers. We observed that resolution Layer 2 received the highest number of requests. The popularity can be attributed to the best trade-off between video details and the field-of-view offered by Layer 2. The *Tracking* tile accounted for the second highest number of requests. Although most users preferred Layer 2 and the *Tracking* tile, other layers are also demanded for serving smaller or larger zoom levels. Figures 8 (c) and 8 (d) show the breakdown of the requests for each tile from Layers 2 and 3. Most requests correspond to the bottom region of Layer 2 and the center region of Layer 3, where the lecturer is most likely to be.

We analyzed the positions and the sizes of the RoIs recorded from five different users watching a 15-minute video segment extracted from one of our lectures. To measure how

often a user operates pan, tilt and zoom, we calculate the non-overlapping area of the user's RoI with the previously requested RoI. Figure 12 plots the non-overlapping area as a ratio of the total RoI area for 3 users over the course of the 15-min video. A high ratio indicates significant change of RoI in terms of its size and position. A ratio of zero represents no change in the RoI. We see that each user changed the RoI at different times, with different amounts, and with different frequencies. Furthermore, given a pair of users, the deviation measure is computed by comparing the respective RoIs of the two users. Also plotted in Fig. 12 is the average of the numbers corresponding to all possible pairs that can be formed with five users. The traces plotted in Fig. 12 confirm that users are interested in watching different regions. When we consider the deviation measure in conjunction with data presented in Figs. 8 and 11, we find that users indeed make use of the interactivity provided by the *ClassX* system.

7. CONCLUSIONS

In this paper, we present *ClassX*, a system for lecture video recording, publishing and streaming. Once a lecture is published online, users can watch it anytime. *ClassX* offers video streaming with pan/tilt/zoom functionality such that users can watch individual RoIs. Despite enabling multiple users to navigate within the recorded field-of-view, it is possible to compress the recorded video content only once and thus avoid on-the-fly video encoding per user. Although several tiles from multiple resolution layers are encoded while preparing the content for online publishing, the complexity of the video encoding is reasonably low. For example, a 75-minute lecture can be published after about 140 minutes of processing.

Besides allowing users to watch arbitrary RoIs of their choice, *ClassX* can automatically create a video that mimics a human camera operator's pan, tilt, and zoom. In the current version of the system, a human operator is only needed to start and stop the recording.

Digital presentation slides discussed during the lecture can be uploaded to the system after the video has been recorded. *ClassX* automatically recognizes which slide is discussed during which segment of the video. The slides can be advanced automatically during video playback. The user can also select a particular slide and jump to the segment in the video where that slide is discussed. It should be noted that the processing time of 140 minutes reported above also includes the time for creating a *Tracking* tile and automatic slide synchronization.

We present and analyze usage patterns that gauge how real users make use of various features of *ClassX*. It was found that users choose the *Tracking* mode as well as control the RoI themselves. The streaming bit-rates are in a range that is feasible with today's broadband speeds.

8. REFERENCES

[1] T. Nagai, "Automated lecture recording system with avchd camcorder and microserver," in *Proc. ACM SIGUCCS fall conference on User services conference (SIGUCCS'09)*, St. Louis, Missouri, USA, 2009, pp. 47–54.

[2] A. Mavlankar, P. Baccichet, D. Varodayan, and B. Girod, "Optimal slice size for streaming regions of high resolution video with virtual pan/tilt/zoom functionality," in *Proc. of 15th European Signal Processing Conference (EUSIPCO'07)*, Poznan, Poland, Sep. 2007.

[3] A. Mavlankar, J. Noh, P. Baccichet, and B. Girod, "Peer-to-peer multicast live video streaming with interactive virtual pan/tilt/zoom functionality," in *Proc. of IEEE Intl. Conf. on Image Processing (ICIP'08)*, USA, Oct. 2008, pp. 2296–2299.

[4] A. Mavlankar and B. Girod, "Background extraction and long-term memory motion-compensated prediction for spatial-random-access-enabled video coding," in *Proc. Picture Coding Symposium (PCS'09)*, USA, May 2009.

[5] Microsoft Silverlight. *Seen on Apr. 18, 2010*. [Online]. Available: http://www.microsoft.com/silverlight/

[6] Adobe Flash. *Seen on Apr. 18, 2010*. [Online]. Available: http://www.adobe.com/products/flashplayer/

[7] S. Mukhopadhyay and B. Smith, "Passive capture and structuring of lectures," in *Proc. 7th ACM Intl. Conf. on Multimedia (Part 1) (MULTIMEDIA'99)*, Orlando, FL, 1999, pp. 477–487.

[8] M. Bianchi, "Automatic video production of lectures using an intelligent and aware environment," in *Proc. 3rd Intl. Conf. on Mobile and ubiquitous multimedia (MUM'04)*, College Park, Maryland, 2004, pp. 117–123.

[9] C. Zhang, Y. Rui, J. Crawford, and L.-W. He, "An automated end-to-end lecture capture and broadcasting system," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 4, no. 1, pp. 1–23, 2008.

[10] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.

[11] Apache HTTP server. *Seen on Apr. 18, 2010*. [Online]. Available: http://httpd.apache.org/

[12] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russell, "Towards robust automatic traffic scene analysis in real-time," in *Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1, Oct. 1994, pp. 126 –131 vol.1.

[13] S. Tsai, D. Chen, J. Singh, and B. Girod, "Rate-efficient, real-time CD cover recognition on a camera-phone," in *Proc. ACM Intl. Conf. on Multimedia*, Vancouver, BC, Canada, Oct. 2008.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, pp. 91–110, 2004.