TalkMiner: A Lecture Webcast Search Engine

John Adcock[1] Matthew Cooper[1] Laurent Denoue[1] Hamed Pirsiavash[2]
Lawrence A. Rowe[1]

[1]FX Palo Alto Laboratory
3400 Hillview Ave.
Palo Alto, CA 94304 USA
{last name}@fxpal.com

[2]Dept. of Computer Science
University of California, Irvine
Irvine, CA 92697 USA
hpirsiav@ics.uci.edu

ABSTRACT

The design and implementation of a search engine for lecture webcasts is described. A searchable text index is created allowing users to locate material within lecture videos found on a variety of websites such as YouTube and Berkeley webcasts. The index is created from words on the presentation slides appearing in the video along with any associated metadata such as the title and abstract when available. The video is analyzed to identify a set of distinct slide images, to which OCR and lexical processes are applied which in turn generate a list of indexable terms.

Several problems were discovered when trying to identify distinct slides in the video stream. For example, picture-in-picture compositing of a speaker and a presentation slide, switching cameras, and slide builds confuse basic frame-differencing algorithms for extracting keyframe slide images. Algorithms are described that improve slide identification.

A prototype system was built to test the algorithms and the utility of the search engine. Users can browse lists of lectures, slides in a specific lecture, or play the lecture video. Over 10,000 lecture videos have been indexed from a variety of sources. A public website will be published in mid 2010 that allows users to experiment with the search engine.

Categories and Subject Descriptors

H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Search,Video

1. INTRODUCTION

Lecture webcasts are readily available on the Internet. These webcasts might be class lectures (e.g., Berkeley Webcast, MIT Open Courseware, etc.), research seminars (e.g., Google Tech Talks, PARC Forums, etc.) or product demonstrations or training. The webcasts typically include presentation slides synchronized with either an audio stream (i.e., podcast) or an audio/video stream.

Conventional web search engines will find a webpage with a lecture on it if you include "webcast" or "lecture" among your search terms (e.g., the search terms "multicore webcast" return several pages from an Intel website that include lecture webcasts), or perform a search on a website specifically oriented towards aggregating lecture content. But users, particularly students, want to find the points when an instructor covers a specific topic in a lecture (e.g., course requirements or discussion about a specific concept). Answering these queries requires a search engine that can analyze the content of the webcast and identify important keywords.

TalkMiner builds a search index from words on the presentation slides in the source video. The system analyzes the video to identify unique slide images. A time code at which the slide is first displayed is associated with each image. At first, we doubted that simple automatic algorithms could reliably identify slides in lecture webcast videos. Image quality in the video stream is usually poor, lighting is unpredictable (e.g., the room could be dark and the projection screen too bright or vice versa), and the projection screen might be located on one side of the room at an extreme angle to the optical axis of the camera. And, some lecture halls have a control booth with production switching equipment to improve the quality of the video. For example, different views of the lecture can be presented by using multiple cameras or a pan/tilt/zoom camera (e.g., speaker close-up, wide-angle stage view, full-screen slide image, audience, etc.). Or, picture-in-picture compositing can be used to produce a video stream with a presentation slide and a small picture of the speaker in a corner.

A slide identification algorithm was developed to detect these video production techniques and correctly identify the slides. The algorithm was tested on lecture webcasts published on YouTube. We were surprised when the algorithm performed better than expected. Consequently, we continued to improve the algorithm and built a lecture webcast search system.

TalkMiner does not maintain a copy of the original video media files. The files are processed to produce metadata about the talk including the video frames containing slides and their time codes, and a search index constructed from the text recovered from those frames. When a user plays a lecture, the video is played from the original website on which the lecture webcast is hosted. As a result, storage

MM'10, October 25–29, 2010, Firenze, Italy.

requirements for the system are modest. The current prototype has indexed over 11,000 talks but it only requires approximately 11 GB of storage (i.e., 1 MB per talk).

This paper describes the design and implementation of the TalkMiner system. It is organized as follows. Section 2 describes the web interface that allows the user to search for and play selected lecture webcasts. Section 3 reviews related work. Section 4 details the slide detection algorithms, the implementation of the off-line lecture processing system, and the implementation of the on-line webcast search system. We also review some experiments validating our slide detection scheme. Section 5 discusses the system including interactive indexing extensions and legal issues encountered when we decided to release the system for public use. And finally, Section 6 concludes the paper with a summary.

Figure 1: Search results page. Lecture videos are shown in a series of rows with a representative keyframe and a title and description when available. The search results can be filtered or ordered based on available metadata including the date, duration, number of slides, and source of the webcast.

2. USER INTERACTION

The TalkMiner search interface resembles typical web search interfaces. The user enters one or more search terms and a list of talks that include those terms in the title, abstract or one of the presentation slides are listed as shown in Figure 1.

The information displayed for each talk on the search results page(s) includes a representative key frame, the title of the lecture, and the channel[1] of the talk. Other metadata displayed includes the duration of the talk, the number of slides, the publication date, and the date it was indexed by TalkMiner. A link is provided that allows the webcast to be played on the original page on which it was published. Notice that search terms are highlighted in green to identify their occurrence.

The user can browse a list of talks and alter the sort criteria for the listing. By default, talks are listed by relevance

[1]For YouTube content the TalkMiner "channel" is the userid of the publisher, or the YouTube channel name for publishers with multiple channels. The "channel" for other TalkMiner content is the publisher (*e.g.*: Berkeley, PARC, etc.).

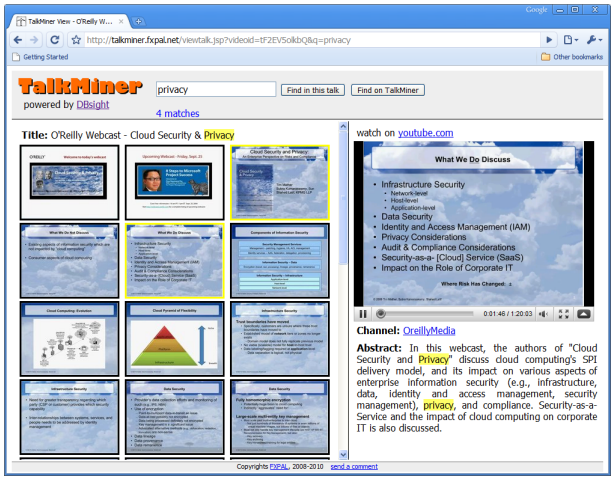

Figure 2: Viewing slides and using them to seek the embedded player.

to the query terms as computed by the Lucene text retrieval software [18]. Other available sort attributes include publication date, number of slides, channel, and rating. The first column on the left side of the results page includes interface controls to filter results according to specific criteria (e.g., the year of publication, the channel, etc.). It also includes a list of recent search queries to allow users to re-execute a recent query. And finally, the interface provides an RSS syndication interface so a user can subscribe to a search result and keep abreast of newly added content that matched the current query with any RSS reader.

Suppose the user wants to browse the third talk listed in Figure 1, which is a talk published on the O'Reilly Media channel on September 30, 2009 entitled "Cloud Security and Privacy." Figure 2 shows the TalkMiner web page for the talk. The left column of the interface displays the title and the speaker's name and affiliation. The rest of the page displays slide images detected in the video. Slides with a yellow border contain one or both of the search keywords, which are highlighted. The user can browse the slides to find one in which he or she is interested. Selecting a keyframe seeks the embedded player, shown on the right, to the segment in the video presentation in which the slide appears, and plays the video.

3. RELATED WORK

Low cost production of a rich media lecture webcast is challenging. Curated collections of academic lecture material can be found on a few websites [25, 21, 2, 27] but the presentation is rarely more advanced than a video with a timeline interface. In cases where an enhanced presentation of the lecture is made available[2], it is the product of authoring and access to the original presentation materials. Many research groups and commercial companies have created applications that produce high quality rich media lecture webcasts, but most solutions impose limitations to simplify content production. For example, some solutions

[2]A subset of the content on videolectures.net is very richly authored with synchronized slide images and talk outline accompanying the embedded video

require offline processing of the presentation video to identify slides and create a search index [24].

Given a source lecture video, the first task is to detect the appearance of presentation slides. Slide time codes can be acquired by manually noting when each slide is displayed. For an automatic solution, time codes can be recorded online by installing a presentation package plug-in to capture media directly from the PC screen or from interfaces to specific presentation software packages [29]. However these solutions fail whenever non-preconfigured PCs, such as a guest lecturer's personal laptop, or unsupported presentation software is used. An alternative solution captures images of the slides by mounting a still image camera in front of the projection screen or by installing an RGB capture device between the presentation PC and the projector [8, 26]. Still images are captured periodically (e.g., once a second) with a time code. Heavyweight solutions that leverage video cameras and room instrumentation can produce rich lecture records, but are notoriously expensive to install, operate, and maintain [1, 20, 5, 19].

Systems that post-process lecture videos or projector image streams commonly use simple frame differencing techniques and assume that the location of slides within video frames is known or fixed and that slides dominate the video frame [5, 20, 8, 26]. Haubold and Kender [10, 11] focus on multi-speaker presentation videos and develop an enhanced multimodal segmentation that leverages the audio stream to detect speaker changes. The use of audio for segmentation has also been studied in [14] for lectures and in more general purpose video retrieval systems [13, 23]. Our work also employs visual frame differencing as a baseline, which we extend with both spatial analysis and speaker appearance modeling to reduce the number of non-slide images in the final set of keyframes.

The next step is to build an index for video retrieval. After segmentation of the lecture video, search indexes are built by associating any detected text with the corresponding temporal video segment. Systems that process audio for segmentation commonly use automatic speech recognition (ASR) output to build a time-stamped text transcript for retrieval [23, 13]. This can be error-prone for low production quality videos such as webcasts for which speaker and vocabulary adaptation may not be possible. [14] uses discourse markers for segmentation and key sentence detection to create a text index. [11] uses text filtering and keyphrase detection to post-process the ASR output for improved indexing. Systems that rely on presentation software APIs or assume availability of presentation files also have a slide text transcript available [29, 15, 16] for indexing. Our work does not currently make use of the audio stream for indexing, although this is a potential direction for extension of our system. Typically, the low quality of recorded audio in generic webcasts results in numerous ASR errors which in turn degrades indexing and retrieval. In such cases, indexing using optical character recognition (OCR) can be preferable [12].

After automatic slide detection, the resulting slide images can be processed to enhance the image and apply OCR and text processing algorithms to create the search index. The ProjectorBox system developed by Denoue *et al.* [8] presented methods for slide capture and retrieval based on inter-frame pixel differences. Vinciarelli and Odobez described techniques for improving OCR for slide retrieval [26] whose performance is competitive with API based methods using the actual slide text. However, both of these systems processed frames directly captured from the presentation projector. Such images are commonly higher resolution than generic webcasts, and include less non-slide content. The system we describe below accommodates generic, low quality production video and increased amounts of non-slide content.

In summary, producing a rich media lecture webcast is complicated and expensive. Solutions that require manual labor are expensive and error prone. Solutions that require installation and operation of special purpose equipment are also expensive and introduce operational problems. The simplest approach to lecture capture, and by far the most widely used capture technology, is a single camera pointing at the speaker and a screen on which presentation material is projected. The question is then how to identify the slides and their time codes, and create a useful text index from such video.

4. SYSTEM DESCRIPTION

This section details the design and implementation of TalkMiner including the system architecture, video indexing, and retrieval interface. An overview of the architecture appears in Figure 3. The system is decomposed into two components: the back-end video indexer and the front-end web server. The back-end video indexer searches the web for lecture webcasts and indexes them. It executes algorithms to identify slide images and post process them to create the keyword search index. The indexing processes currently run on a shared cluster with four virtual machines.

4.1 Aggregating Talks

The system currently indexes lecture videos from four sites: YouTube [28], PARC Forum [22], U.C. Berkeley [3], and blip.tv [4]. The current number of talks indexed is 10,320 with the breakdown of contributions by site shown in Table 1.

Table 1: Source distribution of the indexed content in TalkMiner.

Website	Videos Indexed
YouTube	8104
Webcast Berkeley	1987
PARC Forum	37
blip.tv	192
TOTAL	**10320**

The system processes an average of 29 new videos per day. The method for identifying and accessing the presentation media varies slightly for each platform, but generally videos are identified by parsing RSS feeds to which the system is subscribed. Once downloaded, it takes roughly 6 minutes to process a 60 minute talk (i.e., 10 percent of the real time talk duration), so the system is generally limited by download speed rather than processing speed. Note that the 29 videos per day figure mentioned above is not the capacity of the system. Even with the current modest resources allocated to it, hundreds of videos could be processed per day. The database, which maintains metadata about talks, has a single table with one row per talk. Each row has 20 columns

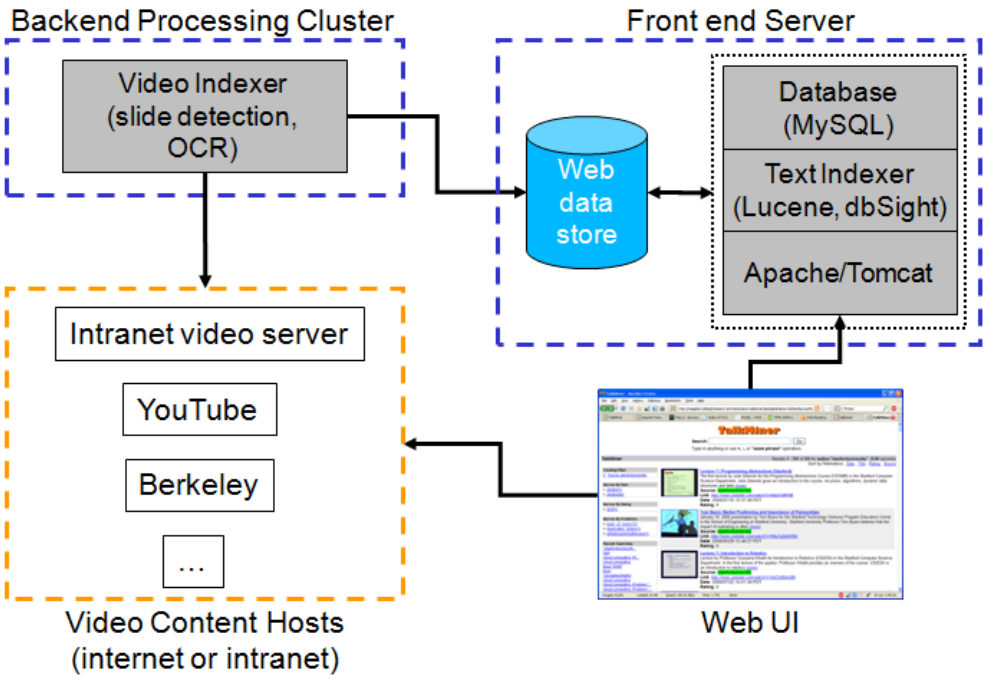

Figure 3: System architecture.

that store information about the talk including the URL to the original media, the title, description, publication date, number of slides, duration, width and height of the video, the date the talk was added to TalkMiner and the date it was indexed. The table also includes a column with the OCR text for all slides in the presentation and an array of offsets into that column where the text for each slide segment begins.

Of the wealth of videos available online, only a relatively small proportion are good candidates for processing and indexing by TalkMiner. Having a human in the loop to identify good lecture sources, for instance the Google Tech talks channel on YouTube, is a very effective way to vet content, but doesn't scale well to include new sources or to incorporate sources that may only sporadically contain lecture content. Currently the bottleneck in the system is downloading the media, which in many cases is not much faster than realtime due to throttling at the source. Because of this, there is a significant incentive to eliminate poorly suited content without investing the time and bandwidth to download the entire media file. Towards this end, we've evaluated a classification strategy that requires only the first few minutes of video.

Using a test set of around 200 lecture videos and 100 non lecture videos, we extracted features from the first 5 minutes of each video to train an SVM classifier to make the binary distinction between lecture and non-lecture material. The features used were: the total duration of stationary content, the number of stationary segments, the average length of stationary segments, and the minimum and average entropy of the projection of vertical edges. The latter measure is intended to be a low-cost indicator of the presence of text. Detection of stationary content was performed using the frame differencing methods described below in Section 4.2. Using a leave-one-out method for evaluation we achieved a 95% classification accuracy. By treating the classification score as a confidence measure and rejecting classifications with low confidence, we achieved 98% classification accuracy with a 9% rejection rate.

4.2 Slide Detection

The principal goal of our analysis is to detect slides and slide changes within lecture videos to extract useful keyframes for user navigation and video indexing. The keyframes are rendered in our web interface to provide users a visual summary as well as entry points into the full video content. We view slide images as ideal keyframes for several reasons. The slides provide immediate context for the lecture content by simple visual inspection without requiring the user to preview any audio or video. Secondly, slides contain text that can often be extracted by OCR and used to generate an index for text-based search into a talk or collection of talks. Lastly, slides are commonly used by lecturers to organize their presentations and thus often delimit topically coherent portions of lectures.

It's worth emphasizing the differences between the slide-based segments that we aim to detect and traditional shot segments. Our goal is to determine the temporal segment associated with the display of a specific slide. In the simplest case, the slide will appear throughout the segment. More generally, a slide segment may contain multiple shots of the (same) slide, possibly interspersed with shots of the speaker, the audience, or other content. The slide keyframes organize both our interface and indexing. Thus, we associate slide keyframes with corresponding temporal segments to facilitate browsing and interaction.

4.2.1 Frame Differencing

We initially adapted the frame difference-based analysis from ProjectorBox [8] to perform basic keyframe extraction system for TalkMiner. We process one frame per second to detect global pixel differences that exceed a threshold relative to the last keyframe. Once 1% of the pixels in the frame exceed the change threshold, a new keyframe is extracted after a period of three seconds during which the global change remains below the threshold. Thus, we extract keyframes that represent only stable video segments, which are more likely to be frames that include slides as opposed to frames containing the speaker, audience, or other content. This basic approach produces satisfactory results for many videos, and works well for videos in which the slides are the predominant focus of the camera. However, we have found it necessary to extend this basic approach to address several frequently occurring cases for which it produces poor sets of keyframes:

- full-frame shots of the speaker that contain neither distinctive visual information for keyframe browsing nor useful text for indexing.
- shots of slides that contain secondary smaller "picture-in-picture" streams, usually focused on the speaker.
- shots of slides from the back of the room that include the audience in the foreground and/or the speaker.

In these cases, basic frame differencing often selects extraneous keyframes that either contain non-slide content or duplicate keyframes of the same slide. In the latter two cases, if the non-slide portions of the frame exhibit continuous motion, then slides can be missed (i.e. not extracted at all) as the scene never stabilizes for three seconds as required in the baseline system.

4.2.2 Incorporating spatial cues

Our first enhancement integrates spatial information into the frame differencing. As before, we sample one frame per second and apply a spatial blur to smooth the frames. We then compute a pixel-based difference and threshold the per-pixel differences to produce a binary image. We threshold the total difference in the center block of the frame using a 5×5 spatial grid, assuming that sufficient change in this block indicates a slide change. This is motivated by the observation that the projector screen on which slides are displayed typically occupies the center of the frame.

We next calculate the bounding box that encompasses the pixel differences exceeding the threshold throughout the frame. If this bounding box is smaller in height and width than a third of the frame's height and width (without occupying much of the frame's center block), it is not selected as a new slide keyframe. Rather it is assumed that some other change in the frame such as an inserted view of the speaker, or speaker or audience motion is responsible for the pixel differences.

This enhancement correctly excludes many of the duplicate keyframes selected using simple frame differencing due to picture-in-picture insertions of secondary streams. These include videos in which scaled down shots of the speaker are overlaid on the RGB stream from the projector. Commonly, these insertions occupy a small portion of the frame, and the bounding box for any motion within them is often even smaller. Also, after masking frame differences with limited spatial extent, the requirement for stable content for at least three seconds in our inter-frame difference analysis no longer mistakenly overlooks slide keyframes due to continuous motion within an inserted, scaled down secondary feed. This was also previously a problem in videos shot from the rear of a larger room in which the audience exhibited motion throughout the display of a slide. In sum, this approach improves both the precision and the recall of our slide detection.

4.2.3 Speaker appearance modeling

To reject spurious keyframes when the speaker appears in all or part of the frame, we learn an appearance model to capture visual characteristics of the speaker and background. We developed two prototypes towards this end, one generative and one discriminative. Both approaches share a training procedure in which a set of 400 frames is sampled from the set of extracted frames that exhibit at least 1% different pixels from the preceding frame. These frames are likely to include the speaker or audience which usually exhibit motion. First, face detection is applied to the sampled frames. For our experiments, we have used the face detection distributed in the OpenCV library. While face detection reliably works for full-frame shots of a speaker, the picture-in-picture or "back of the room" frames may lack sufficient resolution for successful detection. OCR is next applied to the sampled frames to automatically bootstrap sets of slide frames as well as speaker/background (i.e. non-slide) frames. Full frame color histograms are then extracted for each sampled frame. We divide the frame into two horizontal (top and bottom) blocks. In each of the blocks , we compute a 64 bin color histogram. If training sets with at least five images are not automatically determined, we abandon the speaker modeling and retain the full set of keyframes identified using frame differencing with spatial cues as above.

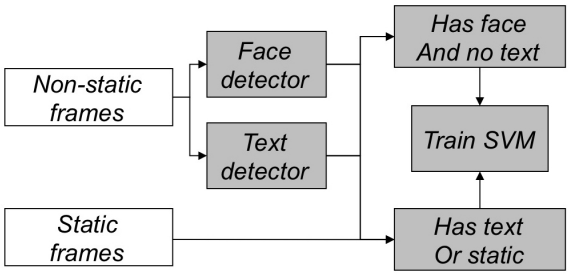

Figure 4: Appearance modeling for keyframe filtering. Static and non-static frames are determined by applying frame motion analysis to a randomly sampled subset of the frames.

In the generative appearance model, the histograms of frames in which faces are detected are averaged to form a template histogram for the speaker frames. This template is compared with the candidate keyframes extracted after frame differencing as above. When the similarity is sufficiently high, the keyframes are deemed full-frame shots of the speaker or background and excluded from the final set of keyframes.

In the discriminative version, we sample 400 video frames and use both face detection and OCR results to group the

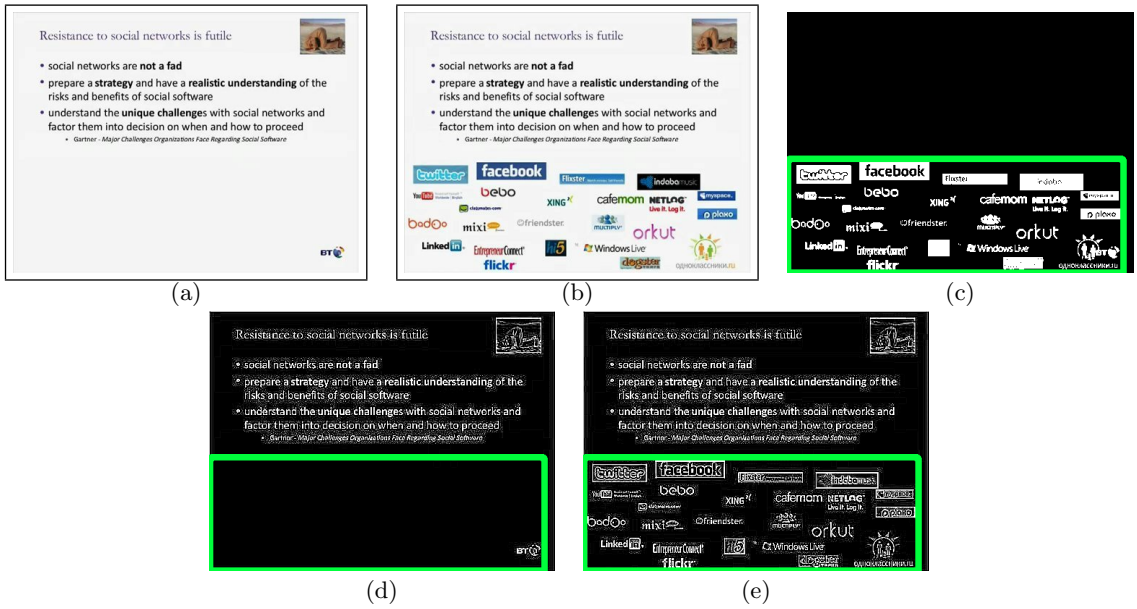

Figure 5: An example build-up sequence detection. (a) shows the first slide in the sequence, (b) shows the second slide with added material. (c) shows the thresholded pixel difference with green bounding box around the changed region. (d) and (e) show the result of edge detection on the original images with overlayed bounding box.

frames as illustrated in Figure 4. Frames with detected faces without detected text are used to model speaker appearance. Frames with detected text and those frames that are stable for over five seconds are used to model slide appearance. These two sets of histograms are then used to train a support vector machine (SVM) classifier to distinguish the speaker/background frames and the slide frames. We use the entire frame to calculate the histogram features. As a result, the appearance of both the speaker and the lecture room or podium area are typically reflected in the speaker training frames. In our experience, the discriminative approach has consistently outperformed the generative approach. Thus, we include only the results using discriminative speaker appearance modeling for keyframe filtering. We discard any candidate keyframes that are classified as members of the speaker/background class.

4.2.4 *Build-up Material*

A fairly common element within presentation videos is the progressive build-up of a complete final slide over sequence of partial versions. An example appears in Figure 5(a) and (b). Frame difference-based slide detection represents each distinct frame in a build-up sequence with a corresponding keyframe. In general, we prefer a more compact representation of a presentation's detected slides, and would rather show a single slide containing all of the content rather than each incomplete version.

We combine a sequence of slides that contain built-up content as illustrated in Figure 5. We first localize the difference between two temporally adjacent candidate slide keyframes. We take the thresholded pixel difference between two keyframes and find the bounding box of any change. This region represents the new content added. If this region in the first slide lacks significant edges, we assume it was empty. If the region with significant edges in the first slide exhibits no edges in the difference frame, then this portion is common content in the two frames. When the regions bounding detected edges in both the difference image and first keyframe mismatch in this manner, we detect elements of a built up sequence. This pairwise analysis is iterated, and the last keyframe of such a sequence is used to represent the entire build-up sequence, creating a more compact visual index without sacrificing any informative content.

By using image analysis to judge whether the change has added content rather than relying on comparing the output of OCR on the two keyframes, we avoid problems with OCR accuracy and also handle the case where the added content is non-textual in nature.

4.2.5 *Keyframe selection experiments*

The keyframe selection algorithm we have developed operates in three phases. The first step is simple pixel-based frame-differencing. We extract frames at 1 Hz from the source video and compute the difference image between temporally adjacent frames and the number of pixels whose difference exceeds a threshold of 24. If the number of changed pixels is greater than 1% of the total, we determine the bounding box of all such pixels and analyze its size and overlap with the frame's center. If the bounding box is either at least a third of the frame or overlapping the center, we detect a new segment. After a period when the global inter-frame difference stabilizes for at least three seconds, we extract a new candidate keyframe as the last frame in the segment.

Next, we sample frames to construct training sets for modeling the speaker/background, and slide images as described above. We extract histograms from the sampled frames and train a SVM to distinguish slide and non-slide frames. We apply the SVM to all candidate keyframes from the first pass and discard any that are deemed non-slide frames. In

Table 2: Experimental results for keyframe selection comparing frame difference-based keyframe selection (Basic) with our extended algorithm (Ext.).

Video ID	Precision			Recall			F1 Score			# Keyframes		
	Basic	Ext.	% Ch.	Basic	Ext.	% Ch.	Basic	Ext.	% Ch.	Basic	Ext.	Ground Truth
1	33.9	64.3	89.7	92.6	92.6	0.0	49.6	75.9	52.9	186.0	98.0	69.0
2	35.9	88.1	145.4	100.0	100.0	0.0	52.8	93.7	77.3	103.0	42.0	38.0
3	40.3	64.2	59.3	86.1	94.4	9.6	54.9	76.4	39.2	77.0	53.0	37.0
4	85.0	85.5	0.6	91.1	94.6	3.8	87.9	89.8	2.1	60.0	62.0	57.0
5	29.0	93.3	221.7	90.0	93.3	3.7	43.9	93.3	112.7	93.0	30.0	31.0
6	58.8	75.9	29.1	95.2	95.2	0.0	72.7	84.5	16.2	102.0	79.0	64.0
7	52.2	77.4	48.3	84.2	84.2	0.0	64.4	80.7	25.2	92.0	62.0	58.0
8	74.2	85.9	15.8	95.8	93.1	-2.8	83.6	89.4	6.8	93.0	78.0	73.0
9	26.1	61.3	134.9	90.2	92.7	2.8	40.5	73.8	82.3	142.0	62.0	42.0
10	73.8	90.4	22.5	95.9	96.9	1.0	83.4	93.5	12.1	126.0	104.0	98.0
11	61.4	86.7	41.2	94.7	91.2	-3.7	74.5	88.9	19.3	88.0	60.0	58.0
12	36.1	75.9	110.2	89.6	91.7	2.3	51.5	83.1	61.4	119.0	58.0	49.0
13	36.8	78.4	113.0	97.7	90.9	-7.0	53.5	84.2	57.5	117.0	51.0	45.0
14	45.2	88.5	95.8	90.4	90.4	0.0	60.3	89.4	48.4	188.0	96.0	95.0
15	62.2	75.4	21.2	93.9	93.9	0.0	74.8	83.6	11.8	74.0	61.0	50.0
16	32.2	50.0	55.3	93.3	100.0	7.2	47.9	66.7	39.2	87.0	60.0	31.0
17	33.9	80.0	136.0	100.0	97.3	-2.7	50.6	87.8	73.4	109.0	45.0	38.0
18	71.4	91.3	27.9	95.2	100.0	5.0	81.6	95.5	17.0	28.0	23.0	22.0
19	50.5	92.9	84.0	91.1	92.9	2.0	65.0	92.9	43.0	101.0	56.0	57.0
20	14.4	57.1	296.5	65.0	100.0	53.8	23.6	72.7	208.3	90.0	35.0	21.0
21	43.4	53.3	22.8	94.5	89.0	-5.8	59.5	66.7	12.1	159.0	122.0	74.0
Avg.	47.5	76.9	84.3	91.7	94.0	3.3	60.8	83.9	48.5	106.4	63.7	52.7

a final pass, we scan the remaining candidate keyframes for any build-up sequences using edge analysis as above. We discard partial versions of any slides in detected build-up sequences, retaining the final complete slide. The remaining set of candidate keyframes are processed for the web interface to the video and text extraction by OCR for indexing. Our indexing code is written in Python, and we use OpenCV for image processing and LibSVM for SVM classification.

To evaluate performance, we manually annotated 21 selected videos from our repository and marked desired slide keyframes and segments. Then, we applied both basic frame-differencing approach of [8] and the extended algorithm above to the videos. We compare the resulting extracted keyframes to the manual ground truth using precision and recall in Table 2. Precision measures the false alarm rate; precision decreases with over-segmentation producing extraneous keyframes. Recall measures the missed detection rate; recall decreases with under-segmentation and undetected keyframes. The F1 score is the geometric mean of the precision and the recall:

$$\text{Precision} = \frac{\#\text{ correctly detected slides}}{\#\text{ detected slides}},$$

$$\text{Recall} = \frac{\#\text{ correctly detected slides}}{\#\text{ ground truth slides}},$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

As noted previously, the main drawbacks of basic frame differencing were duplicate keyframes mistakenly extracted due to intermittent motion and missed slides due to continuous small motion. The results using our extended algorithm show improvement in both precision and recall values, with substantial improvement in precision. The resulting set of automatically extracted keyframes is much closer to the ideal summary represented by the ground truth comprised solely of distinct slide keyframes.

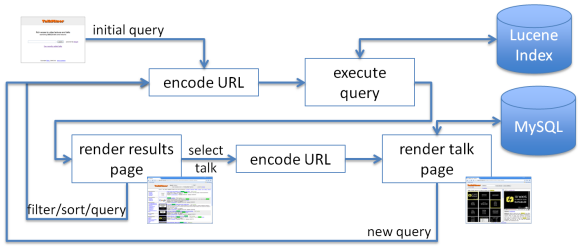

Figure 6: Flow of control and data access through the interactive use of the web site. Search results lists are generated from data stored in the Lucene index and the view of an individual talk is generated with data retrieved from the MySQL database.

4.3 Front End

The TalkMiner web-based front end is implemented in Java server pages (JSP) running on an industry standard Apache/Tomcat combination. The indexing and search framework, implemented with DBSight [7], runs in the same Tomcat instance. At runtime, searches are performed with the DBSight web application from previously computed Lucene indexes of the talk database. At indexing time, the DBSight framework provides a bridge from the database to the Lucene system. Each row or presentation in the database corresponds to a document in the Lucene index, and search operations return a list of presentations. The slides of a presentation are not indexed individually at this level, though below the mechanism for finding a specific slide

matching some search term is described. The data for each presentation is included in the index as some combination of searchable text (title, abstract, OCR text, and channel), a filterable attribute (date, channel), or a sortable attribute (date, duration, #slides). The search results list, depicted in Figure 1, are rendered by customized FreeMarker [9] web page templates. By default the search results are ordered by the Lucene tf-idf text retrieval score, but can be re-ordered on any of the sortable attributes or filtered with any of the filterable attributes at the user's behest. The query, sort order, and filters are all encoded as parameters of the search page URL, making the page completely bookmarkable. Each element in the search results list links to the main talk viewing page for that talk, as shown in Figure 2. This page is a JSP parametrized by unique video id and the query string. The page accesses the MySQL database to retrieve the appropriate row with text and slide timing info for keyframe display and player control. The page thumbnails are rendered with javascript actions to control the embedded flash video player, and slides which match query terms are highlighted. The user can update the terms in the search box to identify slides with other terms. These within-presentation searches are carried out by javascript within the page and don't require any communication with the database. If the user chooses to issue a new search against the full index of talks, the browser is directed back to the search results page once again with query terms encoded in the URL. This flow is depicted in Figure 6.

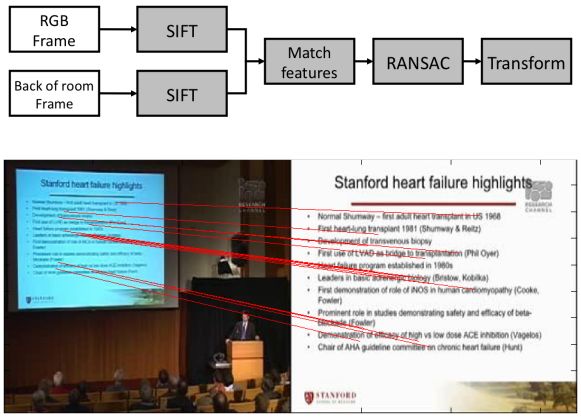

Figure 7: An example of a slide repeated in a back-of-room and RGB capture. Matching feature points are connected by red lines.

5. DISCUSSION

TalkMiner provides a platform for both aggregating publicly available lecture webcasts and searching and browsing webcast content. The ability to search lecture videos by the terms appearing in displayed slides extends search based on metadata and text available on the videos' original web pages. Additionally, identifying such terms with specific slides in a longer source video enables direct navigation to specific video segments of interest in response to text queries.

There are other extensions of the system that can leverage user interaction. Slides can be annotated or tagged by users to provide additional text for indexing. As well, segmentation and keyframing errors can also be corrected by users. While we currently index presentation videos as a unit in our database, our framework can be extended to finer granularities treating each slide as a unit, or grouping sets of temporally adjacent slides. Users can also provide such topic structure by grouping sets of extracted keyframes.

We have experimented with incorporating user interaction to aid in enhancing slide images for OCR in cases when lecture videos are shot from the back of a room. In some cases, this setup significantly degrades the legibility of text in the slides. Knowledge of the screen location within the frame can aid image processing to help enhance the slide image for both OCR and presentation to users. For this we built a simple interface to allow users to manually demarcate the presentation screen's corners within the larger video frame. Given such spatial information, slide images can be appropriately cropped and warped to improve OCR.

When lectures are captured from the back of the room or with a mix of fullscreen RGB capture and cuts to video camera capture, it is desirable to identify when we have captured the same original slide from multiple views. The different views may be RGB capture and back of the room, or multiple back-of-room views resulting from camera motion. To detect this correspondence we have tested a feature point matching algorithm which can identify and align multiple views of the same slide. Given two detected views of a slide, we extract SIFT [17] or similar feature points. As depicted in the top panel of Figure 7, we find matching points and if a sufficient number of points match, estimate a perspective transform between the two images using RANSAC. Given the perspective transform relating the two images we can choose the version of the slide with larger scale as the keyframe to represent the pair.

Given the presentation screen location, we can search for such correspondences using temporal cues. In many settings, lectures occur repeatedly in the same room with a fixed camera. In these cases, the presentation screen's location may be known a priori. When RGB capture is mixed with back of the room video, our speaker/background model usually distinguishes the two feeds readily. Thus if a segment classified as speaker/background is followed by a slide segment (or vice-versa), the matching algorithm can be applied to determine whether the same slide is shown in both segments. The matching also provides a segmentation of the back-of-room camera view. That is, the borders of the full-screen capture map to the borders of the slide in the back-of-room view, providing a segmentation of slide in the back-of-room view. This can be used to crop and rectify the slide region from the back-of-room shot. Using this segmentation it is also possible to create a visual model of the room surrounding the slide and by matching this visual model, detect the identical camera configuration in other portions of the webcast. In these matching camera views the previously estimated slide segmentation and transform can be applied even when there is no corresponding full-screen RGB capture. Note that due to the copyright status on the great majority of the lecture material available to us, we do not perform this sort of cropping and perspective correcting transformation as it would constitute a derivative work.

Video on the web exists under a wide variety of copyright and terms of use. In the implementation of our system we have been mindful of these restrictions with an eye towards making it accessible to the public. In particular, the

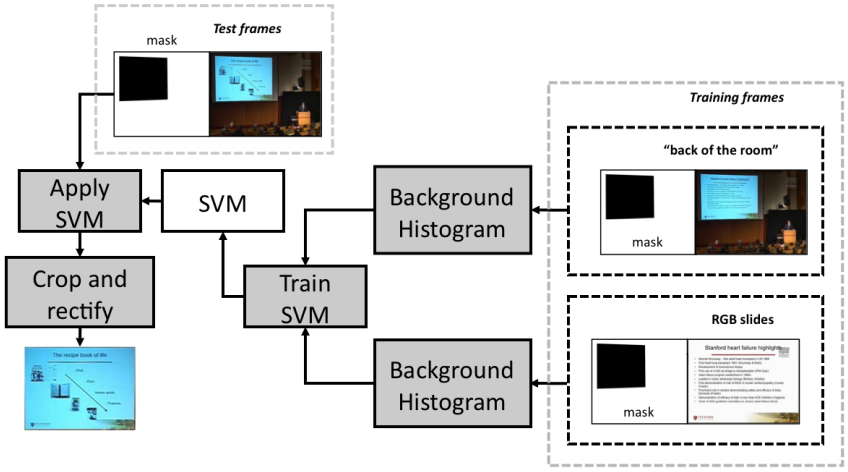

Figure 8: Using a matched back-of-room and full-screen capture to build a background mask and then learn a visual classifier to discriminate full-screen slide views and back-of-room views. In matching back-of-room views the previously determined segmentation can be used to segment and correct the perspective of the slide in the back of room view, even when there is no corresponding full-screen capture of the same slide.

emergence of officially sanctioned embedded flash video is particularly enabling. This allows us to present the original media without creating a copyright violation since redistribution rights are often not explicitly granted. Generally speaking, university lecture material is accompanied with an explicit Creative Commons [6] license, but even these vary significantly in scope. In particular, the creation of "derivative works" is sometimes prohibited which puts certain compelling presentation elements into questionable legal ground. Automatic cropping or color correction of back-of-room scenes as described in Section 5 would generally be considered creation of a derivative work. Likewise other creative re-purposings of the presentations, such as rendering a PDF document of the extracted slides for printing or even displaying a higher resolution rendition of the slides, constitute at least a re-distribution and arguably a derivative work. The potential value of a system like TalkMiner is much higher when using content without copyright restrictions, such as would be the case if the system were deployed by the copyright holder. For instance, if a university were to employ the system on their own archive of lectures.

6. CONCLUSION

TalkMiner is a rich search and browsing system designed to enhance access to and usage of lecture webcasts. The system leverages existing online video distribution infrastructure to embed the original webcast in an interface for efficiently searching and browsing *within* the video. TalkMiner does so with a minimal computational and storage footprint for the indexing system. We have described algorithms which enable robust slide identification in a variety of ad-hoc video capturing scenarios. TalkMiner builds its index and interface from commonly recorded video rather than using dedicated lecture-capture systems, or requiring careful post-capture authoring, or even imposing onerous constraints on the style of the video capture. Thus, the system can scale to include a greater volume and variety of existing and newly created content at a much lower cost than would otherwise be possible. We hope to explore interactive extensions to our system to enhance our indexing and video processing. We have outlined some initial explorations in this area, and expect to update TalkMiner further according to to feedback from users and content creators in response to its public release.

7. REFERENCES

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

[2] Academic Earth. `http://academicearth.org/`.

[3] Berkeley Webcasts. `http://webcast.berkeley.edu/`.

[4] Blip TV. `http://www.blip.tv/`.

[5] P. Chiu, A. Kapuskar, S. Reitmeier, and L. Wilcox. Room with a rear view: Meeting capture in a multimedia conference room. *IEEE MultiMedia*, 7:48–54, 2000.

[6] Creative Commons. `http://creativecommons.org`, 2007.

[7] DBSight. `http://www.dbsight.net/`.

[8] L. Denoue, D. Hilbert, D. Billsus, and M. Cooper. Projectorbox: Seamless presentation capture for classrooms. In *World Conf. on E-Learning in Corporate, Government, Healthcare, and Higher Education*, 2005.

[9] FreeMarker. `http://freemarker.sourceforge.net/`.

[10] A. Haubold and J. R. Kender. Augmented segmentation and visualization for presentation videos. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 51–60, New York, NY, USA, 2005. ACM.

[11] A. Haubold and J. R. Kender. Vast mm: multimedia browser for presentation video. In *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 41–48, New York, NY, USA, 2007. ACM.

[12] A. G. Hauptmann, R. Jin, and T. D. Ng. Multi-modal information retrieval from broadcast video using ocr

and speech recognition. In *JCDL '02: Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 160–161, New York, NY, USA, 2002. ACM.

[13] A. G. Hauptmann and H. D. Wactlar. Indexing and search of multimodal information. In *IEEE Intl. Conf. on Acoustics Speech and Signal Processing*, volume 1, pages 195–198, 1997.

[14] T. Kawahara, M. Hasagawa, K. Shitaoka, T. Kitade, and H. Nanjo. Automatic indexing of lecture presentations using unsupervised learning of presumed discourse markers. *IEEE Trans, on Audio, Speech, and Language Processing*, 12(4):409–419, July 2004.

[15] A. Kushki, M. Ajmal, and K. N. Plataniotis. Hierarchical fuzzy feature similarity combination for presentation slide retrieval. *EURASIP J. Adv. Signal Process*, 2008:1–19, 2008.

[16] G. M. Liew and M.-Y. Kan. Slide image retrieval: a preliminary study. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 359–362, New York, NY, USA, 2008. ACM.

[17] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1150, Washington, DC, USA, 1999. IEEE Computer Society.

[18] Apache Lucene. `http://lucene.apache.org/java/docs/`.

[19] Mediasite by Sonic Foundry. `http://www.sonicfoundry.com/mediasite`.

[20] S. Mukhopadhyay and B. Smith. Passive capture and structuring of lectures. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, pages 477–487, New York, NY, USA, 1999. ACM.

[21] Omnisio. `http://www.omnisio.com/`.

[22] PARC Forum. `http://www.parc.com/events/forum.html`.

[23] D. Ponceleon, A. Amir, S. Srinivasan, T. Syeda-Mahmood, and D. Petkovic. Cuevideo: automated multimedia indexing and retrieval. In *MULTIMEDIA '99: Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, page 199, New York, NY, USA, 1999. ACM.

[24] L. A. Rowe, D. Harley, P. Pletcher, and S. Lawrence. Bibs: A lecture webcasting system. Technical report, Center for Studies in Higher Education University of California, Berkeley, 2001.

[25] VideoLectures.NET. `http://videolectures.net/`.

[26] A. Vinciarelli and J.-M. Odobez. Application of information retrieval technologies to presentation slides. *IEEE Trans. on Multimedia*, 8(5):981–995, 2006.

[27] YouTube.EDU. `http://www.youtube.com/edu`.

[28] YouTube. `http://www.youtube.com/`.

[29] P. Ziewer. Navigational indices in full text search by automated analyses of screen recorded data. In *Proc. E-Learn 2004*, 2004.