Automatic Identification and Storage of Significant Points in a Computer-based Presentation

Paul Dickson, W. Richards Dradion, and Allen Hanson

Department of Computer Science
Computer Science Building
University of Massachusetts
Amherst, MA 01003, USA
{pauld,adrion,hanson}@cs.umass.edu

ABSTRACT

We describe an automatic classroom capture system that detects and records significant (stable) points in lectures by sampling and analyzing a sequence of screen capture frames from a PC used for presentations, application demonstrations, etc. The system uses visual inspection techniques to scan the screen capture stream to identify points to store. Unlike systems that only detect and store slide presentation transitions, this system detects and stores significant frames in any style of computer-based lecture using any program. The system is transparent to the lecturer and requires no software or training. It has been tested extensively on lectures with multiple applications and pen-based annotations and has successfully identified "significant" frames (frames that represent stable events such as a new slide, bullet, figure, inked comment, drawing, code entry, application entry etc.). The system can analyze over 20000 frames and typically identifies and stores about 100 significant frames within minutes of the end of a lecture. A time stamp for each saved frame is recorded and will in the future be used to compile these frames into a jMANIC multimedia record of the class.

Keywords: Automatic Capture, Key Frame Analysis, Video Sampling, MANIC, Automatic Indexing

1. INTRODUCTION

Recently, classroom instruction delivery has moved from traditional didactic lectures toward constructivist techniques (cooperative, active, problem-based, student-centered learning). As lecturer-student interaction evolves, it affects how students obtain a record (e.g., notes) for later study and review. Traditional note taking can conflict with significant class interaction. Notes can be provided in advance by the instructor or captured (and possibly annotated) during classroom activities. Providing notes in advance suggests a more didactic approach using prepared lectures. Capturing classroom activities can come in a number of forms. Current systems for classroom recording range from simple streamed video (Bargeron et al. 1999) to dynamic presentation capture (Anderson et al. 2006) (possibly with shared annotations) to full multimedia records that combine video/audio, enlargements of material presented, table-of-contents, indexes, and possibly integrated searches, chat facilities, note-taking and other tools (Abowd et al. 1996; Abowd 1999; Schapira et al. 2001; Brotherton and Abowd 2004; Ziewer 2006). Methods for creating these classroom records range from hand generated to total automation.

Winer and Cooperstock (2002) report that some students object to prepared lecture notes and captured lecture records, arguing that note taking enhances learning, while other feel liberated. Studies at Microsoft (Bargeron et al. 1999; Tiernan and Grudin 2001) and at Georgia Institute of Technology (Abowd et al. 1996; Abowd 1999; Brotherton and Abowd 2004) report that students are generally satisfied with recorded sessions and make good use of the records for study and review. The ability to add additional post-class notes is a popular feature. More
assessment and experimentation is required, but these preliminary results are promising. When one considers the use of these records for distance-learning, the value increases.

Today, automatic generation of a lecture record appears to be inversely correlated to the richness of features that are associated with the recording: the more feature-rich the presentation, the greater the requirement for hand generation. For a feature rich playback system to be an effective tool, it must have a good index into the significant points in a lecture, that is, significant points must be identified and then have corresponding index points into the synchronized video recording created. Identifying content indexes from written material captured in the recording is extremely difficult. Optical character recognition can be used to pull most text out of what appears on the lecturer’s screen but this is only sufficient when the lecturer is using slides that contain pertinent titles. OCR is not sufficient to create titles when it fails on hand written portions of the screen or when multiple programs appear on screen and it is unclear what piece of the information would qualify as the title.

Some systems have been created that try to bridge the gap between functionality and automation. Often this is accomplished by restricting the lecturer's presentation style to something that can be easily identified. The result usually is close to the automatic capture-low feature end of the spectrum. A good example of this is the Microsoft lecture capture system (Liu et al. 2001; Rui et al. 2003; Rui et al. 2004). It automatically controls camera movement and records a lecture. The lecture is automatically indexxed using slide transitions within the presentation. The system is restrictive in that the lecturer may not use anything other than slides for presentation. Many such systems require the lecturer's slides to be entered independently into the capture system in order to create the index. A number of systems have been developed to automatically capture classroom events, e.g., (Smith et al. 1996; Bianchi 1998; Franklin 1998; Abowd 1999; Mukhopadhyay and Smith 1999; Müller and Ottmann 2000), using more varied delivery modes (web-based lecture notes, record-and-playback systems).

The most full-featured system is Mediasite (Sonic Foundry 2006), which captures the video feed from the presentation computer and uses it to create a visual index. This system works by storing video frames every time a change occurs on screen and using these images as an index. The problem with this system is that it has no method to determine which stored images are significant. Visual indexes also tend to be harder to use than a table of contents. Mediasite is far better at automatic capture than the lecture capture systems mentioned above as it does not restrict lecture presentations as long as they occur on a computer but still fails to gather enough information to create a feature-rich environment.

The opposite of the automated systems described above are systems that use significant manual content authoring. MANIC (Schapira et al. 2001) and its descendants are an example. The MANIC production facility extracted the audio (and later audio/video) and synchronized it with images made from the lecturer's PowerPoint, TeX or other presentation media. A table-of-contents index was hand compiled and linked to video. Simple Boolean search over the text in the slides was provided. As the MANIC framework moved from streaming media to CD media, large portions of the production were automated, but the process still began with a videotape or DVD. Production was reduced to 1–3 production hours per hour of MANIC presentation. Over time, additional functionality such as contextual searches (Kumela et al. 2004), integrated note taking, and collaboration (Ray et al. 2004) was incorporated with in the MANIC browser. The current Java-based version, jMANIC (Wallace et al. 2006), was designed with three goals in mind: cross-platform compatibility (provided by Java), multiple modes of delivery (streamed, downloaded, and CD/DVD), and easy customization. jMANIC uses a plug-in architecture that allows additional functionality to be provided by incorporating a new plug-in, for example extended search, notation, collaboration, etc. The MANIC framework has been applied to courseware, but also to electronic texts, meeting proceedings, etc. The plug-in architecture allows the tool set (beyond replay, table-of-contents navigation, and simple search) to be customized to the application. Although jMANIC offers a flexible and customizable multimedia delivery system, content authoring still takes 1–3 hours per hour of lecture. Our goal is to automate the creation of jMANIC content. To do this we must capture and analyze classroom events from different media streams (PC capture, cameras, other sensors and devices). This paper describes a system that automatically creates a compact visual record of an instructor’s classroom presentation as a small set of selected images (key frames) representing significant points in a computer-based presentation, including but not limited to slides, annotations, and program demonstrations. When compiled into a jMANIC presentation of the class, these key frames will be used to create a table of contents for the presentation. The next steps, not described in this paper, will be to analyze material from the white board in the same manner and to integrate that material with the material captured.
from the computer in order to create a complete table of contents for the lecture. In the Future Work section we will discuss what it will take to create labels for the points within the table of contents.

The system splits the output of the lecturer's PC, sending the signal both to the projector and to a second PC that samples and stores it. Using only VGA input, the system is platform independent. Because no special software is required on his PC, the lecturer requires no training or special preparation. The capture device provides a continuous series of screen captures from the lecturer's PC to the algorithm for visual inspection to determine stable points in the lecture. The algorithm is efficient enough to run at close to real time on a low-end laptop.

This paper includes a survey of related work, a description of the capture hardware and the challenges that result from introduced noise, details of the algorithm, empirical results, and conclusions and future work.

2. RELATED WORK

2.1 Video Capture System

Most work on analyzing presentations from a PC is focused on finding points to be used in creating an index for a record and playback style video (Bianchi 1998; Franklin 1998; Franklin and Flachsbart 1998; Abowd 1999; Franklin et al. 1999; He et al. 1999; Müller and Ottmann 2000; Liu et al. 2001; Ma et al. 2003; Rui et al. 2003; Brotherton and Abowd 2004; Rui et al. 2004; Sonic Foundry 2006; Ziewer 2006). This index allows the user to navigate the recorded material. A far smaller number of systems automatically generate this index and are similar to ours because the index points correspond to significant images that need to be part of a complete visual record and not just slide changes (Abowd 1999; Müller and Ottmann 2000; Brotherton and Abowd 2004; Ziewer 2006). We define significant images as those that differ from images that immediately proceed or follow them and which remain unchanged for a given length of time.

Bellcore Laboratories (now Telcordia) developed the well-known and widely used AutoAuditorium for recording lectures (Bianchi 1998), but this system focuses primarily on camera management. Microsoft, a leader in this area, uses a similar approach with a “virtual director” replacing human operators who selected from among four cameras (Liu et al. 2001; Rui et al. 2003; Rui et al. 2004). It generates an index from slides, requiring a seminar format using only slides to create a visual record, as noted above. The Universität Trier created a system called T-Cube (Ma et al. 2003) that in addition to creating an index also records material written on an electronic white board and integrates it with projected slides.

The eClass system (Abowd 1999; Brotherton and Abowd 2004), is similar to our system in that it allows lecturers to use an electronic white board or a PC. Lecturers can annotate slides and use any program on the eClass classroom computer. The system creates an index, with annotation, from these inputs that is transferred to web pages. This system works well but has major drawbacks: lecturers must convert slides into a special eClass format and store them on the eClass server, electronic white boards tend not to work robustly in all environments, and lecturers are often required to use special pens. Mediasite (Sonic Foundry 2006) has similar functionality to ours with various systems that automatically record a lecture and create a visual thumbnail index. The system allows a lecturer to add annotations as part of the visual index. Mediasite's algorithms determine which frames to capture and store, typically storing a frame every time a change occurs. Unlike our system, small changes and transitions between slides and menus are all stored and used to index into the presentation. Our system ignores “minor” changes. Mediasite requires use of a pre-installed package rather than just a signal splitter and a PC with preexisting software. The TeleTeachingTool (Ziewer 2006), like Mediasite, also allows the user to run any software that is desired and creates a visual index from that material. In addition the TeleTeachingTool allows users to search the lecture material. Unlike our system, the lecturer must log into another machine using Virtual Network Computing (VNC), our system only requires the lecturer to connect their usual computer to a VGA cable. Capture of data is performed on this separate machine. The same VNC system is used by the T-Cube system (Ma et al. 2003).
2.2 Significant Point and Index Generation

Some systems focus on creating indexes and not on recording the video. At Columbia University, Liu and Kender have used Hidden Markov Models to determine scene changes in documentaries (Liu and Kender 2000). This approach works well for finding transition points in video but is more complex than necessary for finding transition points in a lecture. They have also created a system that selects key frames from video recordings of a white board (Liu and Kender 2002). The system finds the last white board image without the lecturer and stores the frame before the white board is erased. This system does not track progression of material over time and thus loses information. Also at Columbia University, Haubold and Kender created a system that selects key frames from lecture videos and groups them (Haubold and Kender 2003) by the medium the lecturer is using, e.g., computer or blackboard. Within these groupings a viewer can click on a key frame and play the associated video clip. This system is more useful for organizing a video than for creating a visual record of events.

Microsoft attempted to automatically create lecture summaries for lectures using PowerPoint (He et al. 1999a). The summaries are generated by ignoring slides that are on screen for less than a specified length of time. The system did not perform well in user studies. Researchers at RICOH developed a system that determines where PowerPoint slides occur in a lecture (Erol et al. 2003) and uses them as an index.

3. DATA CAPTURE AND NOISE REDUCTION

Like all lecture capture systems, ours uses a camera and microphone to create a video of the lecturer. Our system differs from most in that we introduce a VGA splitter between the presentation computer and the projector. The splitter allows a duplicate of the projection signal to be sent to a VGA2USB from Epiphan Systems Inc. (Epiphan Systems 2006). The VGA2USB output is sent to a USB port on a second computer, the capture computer, where the data is stored as individual PNG images. The size of the captured frames matches the screen resolution of the presentation computer. The device typically captures between 2 and 4 frames per second depending on the screen resolution of the presentation computer. The output images typically contain “noise”. Four types of noise are observed: transition capture noise, analog variation noise, shift and fuzziness noise, and signal noise.

3.1 Noise

3.1.1 Transition capture noise

Transition capture noise occurs when the device captures a screen in the midst of a transition (Figure 1). Transition captures are inevitable because VGA2USB runs constantly and is not synchronized to lecture events. It does not occur with every transition but can occur several times per lecture.

Figure 1: An image frame grabbed mid transition between two lecture slides.
3.1.2 Analog variation noise

Analog variation noise reflects variations in the analog VGA signal from the lecturer's PC. Successive captured frames of unchanging information will appear identical to the eye, but the analysis program's pixel-by-pixel comparison often finds minor differences. Typically, over 99% of the pixels in each of the red and blue channels exhibit identical intensities (i.e., the same level in the 0 to 255 range of 8-bit color) in successive, visually identical captures, while 70% of pixels in the green color channel are the same. In visually identical consecutive captured frames, over 99.9% of all pixels across all color channels differ by less than 10 levels. The average difference in a given channel is less than 2 and the standard deviation is less than 1. However, discrepancies in 0.05% of a 1400x1050 pixel frame still yields 735 pixels with significant differences. These analog variations tend to be fairly insignificant; problems are caused when there is a visible difference between frames even though the only change is in shading (Figure 2). This example has an average difference of over 88 for all color channels and a standard deviation of about 15. This type of analog variation occurs to some degree in every lecture.

![Figure 2: Two consecutive frames captured during a lecture.](image)

3.1.3 Shift and fuzziness noise

Shift and fuzziness noise results from certain errors in the analog to digital conversion. Occasionally, spurious pixels are added or legitimate pixels are removed. When this occurs multiple times in a frame everything becomes slightly blurred (fuzzy), although it is usually only noticeable at line edges within the frame. A single large addition or removal shifts the whole frame to the left or right.

3.1.4 Signal noise

Signal noise is likely due to equipment problems. It can appear in any part of the frame as shown in Figure 3. Signal noise tends to last for long sequences of frames; in one captured lecture over half of the lecture frames had signal noise. Signal noise affects pixels in different regions from one frame to the next and occasionally disappears from one frame only to reappear in the next.

![Figure 3: A sample of some of the worst signal noise found within the data sets.](image)

Signal noise does not occur frequently. In our tests, all lectures in which signal noise occurred were captured from a single laptop that appeared to have an incorrectly installed video card. The signal noise may be unique to this computer and therefore not significant to overall results, but signal noise must be expected occasionally.
3.2 Managing Noise

The first pass of the algorithm, described in Section 4.1, is designed to ensure that no frames are stored that include transition noise or fuzziness. It also ensures that analog noise does not cause unnecessary frames to be stored. The second pass of the algorithm, described in Section 4.2, ensures that no shifts cause duplicate frames to be stored. The algorithm is thus designed to handle all types of noise other than signal noise, which needs to be eliminated at its source.

4. THE ALGORITHM

Our goal was to develop an algorithm, implemented in software on a capture computer, to analyze a large number of sampled frames and store only those in which something of significance had occurred, e.g., a slide change, annotations on a slide, or changes in an application such as a pull down menu appearing. As noted, the time at which each significant point occurs is also stored. The algorithm finds these points using three passes (Figure 4). The first pass stores the last stable image preceding a change, a stable image being defined as one proceeded by three or more consecutive, identical frames. The second pass analyzes the output of the first pass and eliminates repetitions. The third pass puts the output into a uniform format.

![Figure 4: Overview of the algorithm.](image)

4.1 First Pass

The first pass of the algorithm runs simultaneously with the screen capture. It determines where changes have occurred by performing a pair-wise comparison of all sequential frames captured, taking the latest frame captured and comparing it, pixel by pixel, with the previous frame as the lecture is being recorded. The pair-wise comparison function counts the number of differences found where the difference is greater than a pre-established threshold. An exact difference cannot be used because of analog noise. We selected a threshold of 100, that is, the color difference must be greater than 100 on scale of 0 to 255 in one of the three color channels, which ensures that even analog noise as great as that shown in Figure 2 will be ignored. How the thresholds were determined is described in Section 4.4.
The pixel-by-pixel comparison ignores the bottom 150 rows from the screen capture because this area is used by the Windows operating system to create pop-up balloons and Mac OS to activate menus. Both would erroneously show large areas of difference.

The number of differences found between two successive frames is divided by the size of the area searched to determine the percentage difference between the frames. This percentage difference is then compared with an empirically determined threshold of 0.002%. The algorithm increments a counter whenever a frame is determined to be identical to the previous one. When a candidate significant transition is identified, the algorithm checks the counter, and if the value is greater than or equal to 3, stores the frame occurring before the transition. If the value is less than 3, the counter is reset to 0 and no frames are stored. In Figure 5, frames $I_{n+3}$ and $I_{m+4}$ are stored but no frame between them is stored.

![Figure 5: Selecting significant images to store.](image)

There are two reasons for the three frame threshold. First, the number of identical consecutive frames is checked to eliminate errors caused by transition or fuzzy noise. These types of noise are constantly changing. By storing only images that persist for three consecutive frames, the algorithm ensures that it does not capture a frame with intermittent noise. Second, the image preceding the most recent candidate significant transition is stored because it will contain the most information. When a lecturer annotates a slide, adds code, or enters data in an application, one sees a series of changes in successive images. By not storing this sequence (e.g., the sequence beginning with $I_{n+4}$ through $I_{m+3}$ in Figure 5), the algorithm does not capture partial changes. Once the lecturer stops writing or entering data, the sequence of images will be unchanging until the slide is changed, new annotations are made, or data are entered. The saved frame will correspond to the last time that a significant state appeared on screen. The image after a transition will become a candidate for storage and its persistence will be tracked with the counter.

This pass of the algorithm runs in real time concurrently with the frame capture but finishes up to 3 seconds after the end of the lecture due to buffering. The total number of frames captured ranges between 2000 and 22000. The first pass typically reduces the stored frames to just over 200, and the second pass further analyzes these.

### 4.2 Second Pass

The second pass of the algorithm takes the results of the first pass and further refines them. The second pass uses the output frames from the first pass as its input and runs pair-wise through its input frames. It also uses a function that determines the number of differences between frames, ignores the bottom 150 rows, and then uses the percentage difference between frames to decide what is significant. The first and second pass differ in important ways.

In the first pass, a function counts all instances when the color difference is greater than 100 within at least one color channel. In the second pass the algorithm makes the same comparison, but when it finds a difference, it searches a 3 pixel by 3 pixel square in the first frame centered on the location of the difference. If an edge is found within the 3x3 square, the difference is ignored, but if an edge is not found the difference is counted. An edge is defined as any pair of horizontal or vertical pixels where a change of greater than 50 occurs in one of the color channels. The purpose of this search is to eliminate errors caused by the shift and fuzziness errors. The overall difference between slide images is great enough that eliminating differences found at edges will not cause a change to fail to register. Only rarely will annotations or data entry highlight an edge, so eliminating differences at edges usually will not affect the differences found.
The second pass uses a threshold of 0.01%. If the difference is greater than the threshold, the first frame of the pair is stored; otherwise no frame is stored. The algorithm also evaluates the contiguity of locations differing between pairs of frames. The algorithm measures this by summing the Euclidean distance between the first differing location and all other identified differing locations. If the sum of the distances is less than 1000 the slides are considered to be identical even if the percentage difference is greater than 0.01%. This technique avoids having transitory events, e.g., a blinking cursor, appear as a slide change.

The second pass preserves consistent timing information by using the same process as the first pass. Because the algorithm saves a frame only when the next one is different, the saved frame corresponds to the last time the captured information appears on screen. The associated temporal information reflects the last time a stable image appeared on the screen of the lecturer's computer.

The second pass always stores the very last frame returned by the first pass. If it is the same as the previous frame, then it is saved because it is a stable image. If it is different from the previous frame it still must be saved because it captures the last information presented in the lecture.

Another difference between the first and second pass is that the second pass of the algorithm runs multiple times. Occasionally a small change occurs just before a major transition, e.g., a slide change. These small changes can result in (almost) identical frames being stored in the first pass. The second pass runs multiple times to eliminate this situation and continues to run until the number of output frames is stable.

The second stage of the algorithm typically takes under 60 seconds in its first pass and about 30 seconds for each additional pass. For most data sets the second stage ran only twice. On one occasion it ran 4 times but on average it ran 2.2 times. On average, 220 frames are input from the first pass and the second pass stores around 100 frames as significant images.

4.3 Third Pass
The third pass of the algorithm simply takes the output images of the second pass and puts them into a standard output format. It does the same with the timing information, putting it into a format that corresponds to the output image names. The formatting change is done to make it easier to compile the results into jMANIC. This pass generally takes under 10 seconds to run.

4.4 Noise Threshold Sensitivity
Both the first and second passes of the algorithm use empirically determined thresholds to separate significant changes from noise. In developing and refining the algorithm, we experimented with a number of heuristics for dynamically determining noise thresholds. Each dynamically determined threshold failed because the noise present in the data closely mimicked changes caused by transitions. Statistical analysis failed to find recognizable patterns in the noise, which exhibited fluctuations from under 10 to over 100 levels in a single color channel. These extreme differences in noise sometimes occurred within the same frames.

This led us to empirically establish thresholds and then determine the sensitivity of the system to threshold changes. We found that the algorithm is not very sensitive to the threshold used to determine pixel noise; only an increase or decrease in the threshold of more than 30 caused retention of noisy slides or omission of real changes. However, the algorithm is sensitive to the threshold for distinguishing differences between successive frames. Changing the threshold used by the first pass (currently 0.002%) affects the number of frames stored. Lowering the threshold causes more small changes (e.g., cursor movement) but fewer intermediate changes (e.g., annotations) to be saved. Raising the second-pass change threshold causes fewer intermediate changes to be saved.

5. **EMPIRICAL RESULTS**
5.1 Lectures
We used VGA2USB to capture class presentations in four university computer science courses taught by 3 different lecturers. The lectures captured were 75 minutes long; the captured video frames consumed up to 6 GB of storage and contained 2000 to 22000 frames. The wide variation resulted from differences in screen resolution and from occasions when a lecturer forgot to plug in the USB cable that started the capture process. The average number frames captured per lecture was 12000.

Seventy-one lectures were captured from the four courses. Frames were projected from Macintosh and Toshiba-Tablet laptops. The lecturers used a combination of slides, slides with annotations, Web browsers, text editors, telnet programs, and occasionally other applications.

The effectiveness of the system was evaluated qualitatively by viewing the key frames captured and determining whether they are significant classroom events, then checking the input to ensure that no significant images were missed. The algorithm runs in close to real time, completing all passes of the algorithm within 2 minutes of the end of lecture.

5.2 Results
The ability of the algorithm to capture key frames may be evaluated by comparing its results to those generated by other systems. Most systems use slides as index points into a lecture and require the lecturer to upload all slides before the lecture. These systems will contain 100% of all slides used and will have no double slides. For lectures that contain only slides our system captured more than 99.8% of significant slide transitions (new line, bullet, animation). Slide transitions were determined by hand annotation of all data collected. Transitions not considered significant include cases where the lecturer moved rapidly through slides to jump from one point in a lecture to another. Such intermediate slides tend to remain on screen for under a second and either are not stored at all or do not persist long enough to be considered stable by the algorithm. The one instance in which a significant transition was missed occurred when a bullet was added and then removed too close to the bottom of the frame. (It occurred in the area masked by the first and second passes of the algorithm to avoid pop-up balloons in Windows XP.) We are looking at ways to handle this rare event. There were no consecutive, identical key frames captured by the system. Identical key frames were captured when a lecturer revisited a previously viewed slide. These results are equivalent to those of systems that use only slides to index. These results were found for all slide based lectures that did not include signal noise caused by bad cable connections.

It is more difficult to assess the quality of the key frames captured from non-slide based lectures. The eClass system (Abowd 1999; Brotherton and Abowd 2004) captures every URL visited by a browser as well as all notes made on a tablet surface and stores this information onto a web page. A viewer can use the list of slides shown, notes made, and web sites visited as an index into the saved data. Our system also captured 100% of all URLs visited within the lectures and all tablet annotations. Our system has the benefit of storing intermediate annotations that show the progression of the annotations while eClass has the benefit of a more concise set of index points. Unlike eClass, our system can handle any program or event and does not require the lecturer to use special computers.

The system most comparable to ours for data capture is Mediasite (Sonic Foundry 2006). Mediasite captures every change that occurs on the lecturer's screen in order to create a visual index. Therefore, like our system, Mediasite can handle any program or event used and is transparent to the user. Unlike Mediasite, our system stores only those key frames where something “significant” occurs. While Mediasite stores a key frame for every transition, our system stores only those that are stable, that is, that remain on screen long enough for the viewers to examine them. The only adjustment that can be made to Mediasite is an adjustment as to how sensitive it is to the changes it uses to determine when to store key frames.

The TeleTeachingTool (TTT) is the system most similar to ours in methodology (Ziewer 2006). TTT is able to identify and store significant index points and to use these points to create an index. Like Mediasite, TTT can handle any program that is used on the lecturer’s computer, but unlike Mediasite and our system, TTT requires the lecturer to bring up their desktop on another machine using VNC, adding to the number of tasks the lecturer must perform in order to begin a lecture.
A quantitative measure of the generated sequence of key frames is not possible as there is no quantitative definition of “significant” within the context of key frames. As such only a qualitative analysis of results is possible. The system successfully captured a key frame for 100% of all annotations that appeared on screen for at least 2 seconds as determined by hand annotation of all of the input data. It also captured intermediate key frames that occurred while the annotation was being written (Figure 6). Whether each intermediate step of the annotation is significant is open to debate. The authors intend to answer this question and determine the quality of these results through future user studies.

![Figure 6: The two images are a pair of stored consecutive key frames stored mid annotations.](image)

The system also captured key frames from data entry and program demonstrations. During input to a program, the system captured intermediate steps when the lecturer stopped typing to discuss a point (Figure 7). Similarly, when the lecturer paused a program demonstration to ensure that viewers saw each step, the system stored key frames. The pop-up menu appearing in Figure 7 is an example. Whether all intermediate saved key frames are necessary will also need to be determined through user study.

![Figure 7: The image on the left is from a code implementation during a lecture while the image of the right is from an application demonstration.](image)

While the system stored no two consecutive, identical key frames it did occasionally store key frames that were almost identical. This occurred when a combination of minor changes, such as a cursor movement concurrent with a change in on-screen date/timer, caused the algorithm to store an extra key frame. Also the capture device occasionally produced a multiple pixel image shift resulting in two key frames being stored that were identical except for the shift. Such key frames are hard for the system to handle but are not considered significant as they make up only 0.2% of the key frames stored. Currently the system fails to capture highly dynamic elements, e.g., video clips, animations. We are investigating how to handle these or whether, for note-taking purposes, they need to be captured at all.
6. FUTURE WORK

We intend to continue the process of automating the creation of jMANIC (Wallace et al. 2006) presentations. We have begun to equip a lecture room with computers and cameras that allow us to capture all that occurs on the board at the front of the room. We are developing algorithms that allow us to track the lecturer to create the Lecturer Video window found in jMANIC. We are also developing methods of isolating text written on the board and determining what segments of text correspond to each other. We then intend to integrate the isolated text segments with the sequence of stable images captured by the computer to be used in the Lecture Material window in jMANIC. The timing information regarding stable images will be used to create the jMANIC table of contents. This process will remain transparent to the lecturer who will merely walk into the room, strap on a wireless microphone and lecture. Once these tasks have been accomplished attempts will begin to perform optical character recognition on the captured images and text from the board in order to provide labels for the table of contents instead of just times.

7. CONCLUSIONS

Our algorithm is able to capture and store images to create a complete, accurate and compact visual record of computer-based classroom presentations. The system supports all PC platforms and operating systems, and a wide range of teaching and presentation styles. The system stores a small set of key frames that reflect significant classroom events and eliminates transitory events (cursor movements, system events), noise, and navigational transitions. It is efficient, completing all analysis within minutes of the end of the lecture. The system stores the set of key frames with timing information that allows the creation of Web-based notes and a variety of record and playback style formats. Importantly, the system only requires that instructors connect a PC a projector cable as they normally would. The capture system can be installed on a low-end laptop or on a PC in the classroom.

REFERENCES


