Navigational Indices and Content Interlinkage On The Fly

Peter Ziewer
Institut für Informatik, I2
Technische Universität München, Germany
ziewer@in.tum.de

Abstract

Lecture recording provides learning material for local and distance education. The TeleTeachingTool uses the most flexible screen recording technique to capture virtually any material displayed during a presentation. With its built-in annotation system teachers can add freehand notes and emphasize important parts. Unlike other screen recorders, our implementation offers slide-based navigation, full text search and annotated scripts, which is obtained by automated post-production. This paper presents how to achieve live access to any slide with corresponding annotations created earlier during the presentation, although annotations are not associated with slides. For this the post-processing algorithms must be adapted to work on the fly while the presentation is still in progress and an interlinkage between slides and annotations must be achieved.

1. Introduction

Lecture Recording is conserving presentations for later playback and provides additional learning material, which can be created rather cheaply and quickly and therefore is called *lightweight content creation* [1]. Lecture recorders based upon the screen recording technique, like *Camtasia* (TechSmith Corporation) or our *TeleTeachingTool*, digitally grab the content of the presentation machine's desktop on a pixel basis and therefore are very flexible as they can capture any application (e.g. presentation software and browser) including pointer movements, animations and annotations (e.g. notes and sketches drawn with an electronic pen). However, this flexible technique is criticized for omitting document structures and textual content [2], which provide useful and, as postulated by [1], necessary advanced playback features such as slide-based navigation and full text search. We have shown in [10] how to regain document structure and achieve the required features by automated analysis of the pixel-based recordings. Thus we diminished the major drawback of screen recorders when compared to recorders using symbolic representation, e.g. *Authoring on the Fly* (AOF) (University of Freiburg) or its commercial successor *Lecturnity* (imc AG), which store structured documents, but are rather limited in the choice of the presentation software and supported document types.

Additionally we have introduced a simple but effective annotation system for the TeleTeachingTool to enrich the learning material by adding freehand notes and highlighting important elements during the presentation [9]. Annotations are not bound to the presentation software, but are applied to the desktop as a whole, which makes them usable for all applications. Unfortunately, annotations will not be redisplayed if accessing a slide that was annotated earlier during the presentation. Manually or automatically (as occurs when switching to another slide) deleted annotations are lost and cannot be replayed later (except for an instant undo) because annotations are independent of the underlying content and therefore are not associated with slides. However, such association is asked for by [2].

In this work we will show how to achieve live access to annotated slides by adapting the automated analysis, originally designed for post-production only, to be performed on the fly while the presentation is still in progress, and how to interlink annotations not only with indices but also with content (mainly representing slides). Furthermore, we discuss how to replay earlier annotated slides. We will start with a short overview of the environment used and previously published research results that are the basis for this work.

2. The TeleTeachingTool Environment

During our research we have implemented the TeleTeachingTool (TTT), a freely available, cross-platform lecture recording and broadcasting environ-

ment, which offers flexible screen recording (enhanced with audio and video). It supports various operating systems and allows the parallel use of arbitrary applications, including the teacher's choice of presentation software, animations and browsers. The recorder can be seamlessly integrated into an existing teaching environment in a transparent way without influencing the teacher [8]. Unlike other screen recorders, the TTT offers slide-based navigation and full text search.

Furthermore, the TTT includes a simple but effective dynamic annotation system offering the possibility of drawing freehand notes and sketches as well as emphasizing presentation content by highlighting or underlining. All annotations are recorded and will be replayed dynamically at the appropriate time during playback. In conjunction with the built-in whiteboard, which provides a plain page on demand, the freehand drawing feature replaces the usage of common blackboards or overhead projectors in order to be recorded digitally. Figure 1 shows the TTT displaying an annotated slide and thumbnails for slide-based navigation.

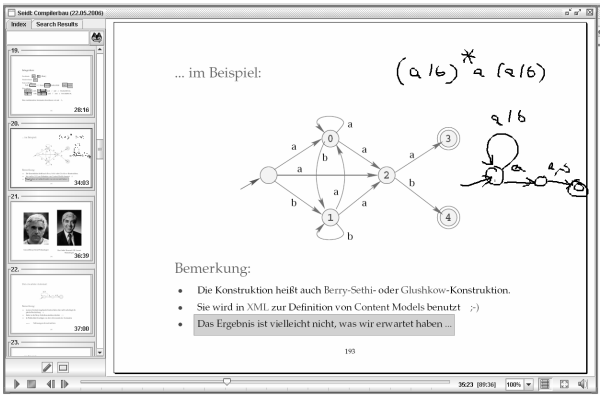

Figure 1. TeleTeachingTool

In order to digitally grab and record the teacher's graphical desktop the TTT uses *Virtual Network Computing* (VNC) [6], which is a remote display system, that allows a graphical desktop environment to be controlled from anywhere on the internet. The technology underlying the VNC system is a simple protocol for remote access to a graphical user interface, the *Remote Framebuffer* (RFB) protocol [7]. It works at the framebuffer level and thus is totally independent of operating/windowing systems and applications. A framebuffer update message represents a change from one valid framebuffer state to another and contains rectangles of encoded pixel data to be placed at a given x,y position. Recording is achieved by logging timestamped RFB protocol messages for later playback [3, 4]. The TTT uses the timestamps to synchronize playback with audio/video streams. The RFB messages received from a VNC server are slightly adapted by the TTT to enable direct access to rectangle headers and to facilitate the reading or skipping of complete messages during post-processing and playback. Furthermore, additional message types are introduced to integrate annotations. Further information, the TTT software and our lecture archive are available at http://ttt.uni-trier.de.

3. Screen Recording: Flexible & Structured

The screen recording technique is known to be very flexible as it can capture virtually any material displayed during a presentation, but unfortunately disregards document structure and textual content needed for slide-based navigation and full text search. Our core idea to regain structure (described in [10]) is to automatically create indices as access points to certain positions within the timeline of a recording, which allows it to be subdivided and thereby structured. Meaningful structuring is automatically gained by *derived* and *side-effect indices* only. Manual structuring by placing *intentional* or *post-hoc indices* is discouraged as this would conflict with a lightweight production process. A detailed index classification is given in [5, 4].

3.1. Slide and Animation Detection

Navigation by slide is one main feature a playback engine should offer [2]. A slide is (mainly) an image shown to the audience. In the case of a VNC-based screen recorder, switching slides during a presentation results in huge framebuffer update messages. The timestamps of updates, which cover a suitably large area of the framebuffer, are potential indices for slide changes. In contrast to the original update messages of the RFB protocol the adapted messages used by the TTT offer direct access to rectangle headers, which are needed to calculate the affected area without parsing the complete message. Rectangles belonging together are identified by their identical timestamps. It is useful not only to combine framebuffer updates assigned to exactly the same timestamp, but also those included in a short period because updates may be split and transmitted with a short delay. Calculating the covered areas and comparing them with an empirically acquired threshold delivers indices of possible slide changes.

The indices obtained by this method provide a partitioning of the recording, but not all of those indices are equally meaningful. A classification is obtained by analyzing the set of indices. Indices that are followed by a further index after only a short delay may not be that important. Analyzing not only the timeshifts between indices but also the length of a sequence of indices with

small timeshifts allows indices to be classified, distinguishing between animations (long sequence) and skipped slides (short sequence). The slide detection can be further improved by using annotation events, especially the event that removes all current annotations, which can be generated either manually by the teacher or automatically by the TTT recorder based upon dedicated key events used to switch slides (e.g. page up, page down). Those events can help to distinguish slides and animations, as animations typically are free of such events, and can be used to gain a more fine grained partitioning of slides whenever the teacher manually clears previously made annotations within a slide.

During playback, slide indices are presented as thumbnail overview and clicking on a certain preview image causes an instantaneous playback of the corresponding slide. Besides providing slide-based navigation, the indices are also used to automatically generate screenshots to which optical character recognition is applied in order to create a search base for full text search and an HTML script containing linked screenshots. The generated script will, unlike published slides, include the annotations made during the presentation and makes them available to students.

3.2. Slide and Animation Detection On The Fly

The automated post-processing consists of 3 phases:

1. derive potential indices
2. classify indices
3. compute thumbnails / screenshots

The first phase, which computes the areas covered by each framebuffer update, can be transferred to online processing in a straightforward manner. As each message must be parsed by the TTT recorder for recording and transmission purposes anyway, each rectangle header is read and the affected area can be calculated. Combining updates of almost identical timestamps (as is done during post-processing) causes a short delay of several hundred milliseconds only.

In the second phase the identified potential indices are classified to determine a meaningful selection. The offline classification algorithm analyzes sequences of indices following each other in short time spans. However, if messages are received from a stream instead of being available at the start, it is not always evident where such sequences end. The empirically determined properties of our current implementation distinguish (supposed) animations after 30 seconds. A slightly delayed index computation is acceptable, because immediate usage of newly created back references is not very reasonable. They either refer to the currently displayed slide or to one which was shown recently for a very short time span only and hence cannot be very important and will contain only a few, if any, annotations.

A sequence that ends within 30 seconds is assumed to be caused by skipped slides, opening a new application window or delayed messages due to heavy server or network load and thus the last index is rated to be valuable. A sequence is classified to be (part of) an animation, if it exceeds the length of 30 seconds. Animations can last longer, but only the start point and thus the first index is classified as important. As soon as the algorithm assumes that an animation is in progress, it can drop any potential indices (exceeding the area threshold) until the end of the animation is detected by a decreasing rate of framebuffer updates. Keeping in mind that we want to allow teachers to access annotated slides during the presentation, it is doubtful whether past animations should be accessible at all. If not, animations are simply ignored. They could also be treated as short sequences, meaning that the last index in the sequence will be classified as suitable and thus the final framebuffer at the end of an animation with all annotations would be accessible. However, as it is unsolved how to guess the content and intention of an animation, it is not possible to determine if a suitable snapshot should be achieved at the entry point, the end or any position somewhere in between.

A thumbnail overview (Figure 1) is a meaningful representation of indices. It can be updated dynamically whenever a new index is detected or an existing one should be replaced due to a higher classified index. As a perpetually changing index overview may confuse the teacher, replacing should be reduced to a minimum. This is achieved by delayed updating, which perfectly fits with the delayed index detection described above. However, the teacher might expect an instant feedback whenever showing a new slide. Therefore a new thumbnail is added immediately whenever the teacher switches to the next slide, but any potential indices appearing shortly afterwards are not displayed until fully classified. This gives an immediate feedback during an ordinary presentation (with an adequate amount of time between slides), but does not confuse presenters due to bustling activity in the thumbnail index during animations or while skipping slides. Similarly, annotations are gathered and added to the corresponding thumbnail with a little delay, because there is no necessity to display them immediately as they are also visible in the main window and the teacher is obviously still occupied with annotating the current slide. As the intention is to access previously made annotations it is advisable not only to index slide changes, but also to make use of side-effect indices [4], caused by deleting all annotations. As result each slide can have several

sets of annotations, which refer to different remarks of the teacher and can be accessed individually.

In order to create thumbnails the post-processing algorithm computes screenshots by fast replay of all update messages and copying the framebuffer's content for each timestamp that represents an index. The online algorithm must store screenshots during index computation, because the framebuffer is modified by every update and reclaiming its content demands the session to be in memory and the usage of a second framebuffer, which is inefficient. Therefore screenshots are stored for each potential index and deleted if the index is rejected afterwards. To avoid performance problems caused by storing many screenshots within a few seconds, it is advisable not to store a screenshot immediately after receiving a potential index, but to wait until the next framebuffer message arrives. This offers the possibility of observing the next header, which may reveal that the new message should be included in the screenshot due to an identical timestamp, or alternatively may result in the generation of a more suitable index to replace the current one. However, as an update can contain several rectangles but the RFB protocol does not allow access to rectangle headers without parsing all preceding rectangles, either only the first rectangle can be observed or rectangles must be parsed and buffered but not immediately displayed. Screenshot generation is reduced further whenever the detection algorithm has identified the currently read sequence as an animation and thus all potential indices can be ignored until the end of the sequence is determined.

3.3 Live Replay

During offline playback a recorded presentation is replayed dynamically in the same way as it was presented in the lecture hall including the teacher's verbal narration. The narration is obviously not needed if accessing a previous index during a live lecture. Dynamic replay of recorded application usage may be meaningful to show the behavior of an application again. However, in most cases it is likely to be easier and less confusing to rerun the application once more instead of replaying the recorded version, because replaying does not allow interaction with the recorded applications and the index might not refer exactly to the position the teacher had in mind. Additionally, if accessing an annotated slide, teachers expect annotation to be displayed instantaneously rather than to appear after a while. Furthermore, dynamic replay demands to keep the whole recorded session in memory, because reading from a file while still recording to it is error-prone and also the replay itself must be recorded again.

In order to represent earlier annotated slides the much easier approach of displaying static screenshots is more suitable. Even editing TTT annotations is possible, because they are handled on a separate layer. Regarding the results of the classification algorithm allows annotated screenshots to be shown if indices are classified as slides, and the dynamic replay of animations or other content elsewise.

4. Interlinkage of Annotations and Slides

TTT annotations are not bound to the presentation software but are applied to the desktop as a whole and hence are applicable to any kind of application. On the other hand, [2] postulates annotations should be associated with slides so that annotations disappear when a slide is changed and made visible again when returning to that slide later during presentation. The first aspect is solved by applying automated removal of annotations triggered by the keys commonly used to switch slides.

Annotations can be linked to indices according to their timestamps. An interlinkage to indices (or any other timestamps) can be achieved by aggregating all annotations in the period between two subsequent indices (or a timestamp and the next event that deletes annotations). In the offline case this approach is suitable to gain annotations that are valid at keyframes and to create annotated screenshots during the automated HTML script generation process. On the fly interlinkage of annotations and already computed indices is easily achieved by buffering annotation events. Accessing a slide via thumbnail overview redisplays previously made annotations. However, this is only a loose interlinkage, because indices are only referencing slide changes without any knowledge of slide content. A slide shown twice during presentation causes two independent slide indices. A real association between annotations and slides (or any other application's contents) would even allow recalling corresponding annotations when skipping back to previously annotated slides within the presentation software.

4.1. Content Interlinkage

In order to achieve real interlinkage it is necessary to determine if the currently displayed framebuffer matches any previously shown content. Comparing the current framebuffer with all previous states is not applicable as every update message modifies the framebuffer content and a session of 90 min. comprises of several thousand updates. However, only grave modifications are important such as switching to another slide or opening a new application. But those are identified

by the indexing algorithm and typically limited to several dozen occurrences. Therefore framebuffer comparisons are only needed whenever a received update message is identified as a potential index and the number of comparison partners is limited to the already identified indices. Detecting exact matches using checksums (such as CRC32) is a relatively easy task. Different checksums unfold different framebuffers and, if chosen suitably, matching checksums should point to equal content at high probability.

Unfortunately, such comparison will be problematic unless contents match perfectly, which is not necessarily the case. Sources of inaccuracy are animated banners of web pages or a clock displayed within an application or the task bar. Also the frequently changing pointer position is a disruptive factor (if part of the framebuffer and not treated separately by VNC). TTT's own annotations are stored on a separate layer, but annotations generated by any presentation software influence the framebuffer as well. Therefore only a high degree of covering instead of a perfect match should be used as the comparison factor. Examinations of the computer science course *Informatik III* (Winter 2005/06) of Prof. Schlichter (25 recordings of approx. 90 min.), revealed a threshold of 1.1% differing pixels as suitable to determine identity of slides. Applying the same threshold to recordings of the courses *Compilerbau* and *Abstrakte Maschinen* (both Summer 2006) of Prof. Seidl showed less perfect matches due to the heavy usage of slide overlays during the presentations. Such overlays are very similar as they partly contain the same content, but nevertheless should be distinguished. Lowering the threshold to a value below 0.2% eliminated the problem. Surveying several other recordings confirmed the lower threshold to be suitable for most lectures. However the detection rate for the lectures of Prof. Schlichter is remarkably better when the higher value is applied due to the presentation environment used, which is not any designated presentation software but rather a web browser showing HTML-based slides. Navigation is done via links and followed links are displayed in another color, which is the reason for the higher number of differing pixel values. Until further research exposes an adaptive threshold computation, a preset suitable for most cases but adjustable for special occurrences is practicable. At least a threshold stays valid for a certain presentation style and thus have to be designated only once per teacher or lecture series. Lowering the color depth before performing comparisons can also reduce irritations.

Pixel-based comparison of several framebuffer contents is not very efficient due to the heavy memory usage and the high number of comparison operations required. However the number of effective pixel values is very limited as for most slides approximately 95% are background color (assuming a single colored background; see Figure 2). Even very complex slides rarely contain over 35% of pixels not colored as background. This leads to very high compression rates even for simple and therefore fast compression schemes such as run-length-encoding. As the vast majority of comparison partners represent unequal slides, the comparison algorithm should detect and reject them as fast as possible, at best by a single value comparison. Examination of several dozen recordings exposed the number of background pixels to be a suitable criterion. Slides cannot match each other if the difference in background pixels exceeds the previously mentioned thresholds of 0.2% or 1.1%, because differing background pixels are a subset of all differing pixels. The complete comparison of all pixel values must be carried out only if the number of background pixels almost matches. Note that quick rejection fails if color cycling or background images are used instead of a solid background, but their usage for TTT/VNC is discouraged due to bad compression ratio anyway.

Constructing a color histogram to identify the background color and to count pixels requires each pixel to be accessed once, but that is also the case for any other comparison algorithm. At least it can be combined with constructing additional data structures to perform an efficient pixel comparison, if needed. However, the main case of comparing non-matching framebuffer contents can be achieved in a time of O(framebuffer width x height) to create the histogram plus a negligible number of indices single value comparisons instead of O(number of indices x width x height) pixel comparisons. Another approach could be a similarity hash, but a suitable hash function needs to be ascertained first.

4.2 Content Prediction by Color Histograms

Through the examination of the background color of recorded lectures we have detected that the color histogram exposes information about the framebuffer content. For simple slides over 90% of the pixels are in background color, but more complex slides achieve values of approximately 55-85%. A desktop with a taskbar, icons and windows results in a coverage of 30-50% in the most frequently used color and if no color covers more than 5% of the pixel values, the framebuffer represents a fullscreen video or high colored picture. Surveying the second most frequently used color of complex slides reveals that a value of more than 10% indicates a table or diagram, but lower values most probably point to a slide containing a high col-

ored picture (e.g. a photo). Figure 2 displays the analysis of one exemplary recording (*Informatik II,* 04/15/2005 by Prof. Seidl). It shows how much of the framebuffer is covered by the most frequently used color for each index. During the corresponding lecture the desktop with some windows was visible at the beginning (up to index no. 13) and the end (no. 50-54). The middle part consisted of a slide presentation and some slides contained images (no. 19, 20, 23, 40 and 41). Index no. 24 was a plain whiteboard page and thus resulted in 100% background pixels. Note that all mentioned recordings are available at our lecture archive.

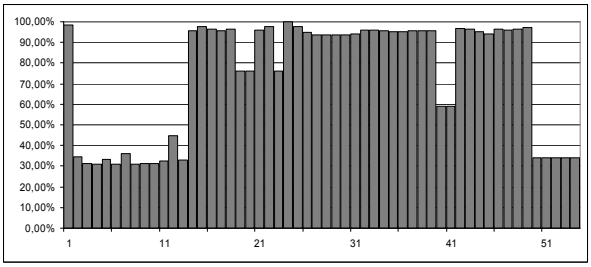

Figure 2. Background coverage

5. Conclusion

Screen recording offers a flexible technique for lecture recording as it allows virtually any material displayed during a presentation to be captured. Automated analysis compensates for many drawbacks caused by the missing structure or symbolic representation of content. Our TeleTeachingTool is a VNC-based screen recorder, which offers automated slide detection to generate useful navigational indices. The post-processing algorithm was transferred to on the fly usage to achieve live access to previously annotated slides, which is requested as useful feature by [2]. By comparing framebuffer contents, annotations can even be linked to slides (or other framebuffer content). The process can be further improved by developing an adaptive threshold computation in future.

The easy to use navigation via a graphical overview of slide indices, which was previously available for later playback only, can also be adapted for online usage while presentation is in progress. The thumbnails (representing indices) and annotations are updated as required by the index detection, but with a short delay, which is sufficient to avoid irritations resulting from a permanently changing thumbnail overview. As we assume that full text search is rarely used during live presentation, and because optical character recognition is a complex task, this feature remains for post-production and playback usage.

An examination of the background color of recorded lectures has revealed that color histograms give information about content, but more research is needed to achieve suitable thresholds for content prediction and to integrate appropriate search and navigational features in a reasonable way. Furthermore, analyzes of dynamical content should be improved.

References

[1] P.-Th. Kandzia, G. Kraus, and Th. Ottmann, "Der Universitäre Lehrverbund Informatik - eine Bilanz", *Softwaretechnik-Trends 24:1*, Gesellschaft für Informatik, Feb. 2004, pp. 54-61.

[2] T. Lauer, and Th. Ottmann, "Means and Methods in Automatic Courseware Production: Experience and Technical Challenges", *World Conference on E-Learning (E-Learn'02)*, Montreal, Canada, Oct. 2002.

[3] S. Li, Q. Stafford-Fraser, and A. Hopper, "Frame-buffer on demand: Applications of stateless client systems in web-based learning", *5th Int. Conference on Information Systems Analysis and Synthesis (ISAS'99)*, Orlando, FL., 1999.

[4] S. Li, M. Spiteri, J. Bates, and A. Hopper, "Capturing and Indexing Computer-based Activities With Virtual Network Computing", *ACM Symposium on Applied Computing*, Como, Italy, 2000, (2), pp. 601-603.

[5] S. Minneman, S. Harrison, B. Janssen, G. Kurtenbach, T. Moran, I. Smith, and W. van Melle, "A Confederation of Tools for Capturing and Accessing Collaborative Activity", *ACM Multimedia'95*, San Francisco, CA, Nov. 1995, pp. 523-534.

[6] T. Richardson, Q. Stafford-Fraser, K. Wood, and A. Hopper, "Virtual Network Computing", *IEEE Internet Computing*, 1998, 2 (1), pp. 33-38.

[7] T. Richardson, "The RFB protocol. Version 3.8.", *http://www.realvnc.com/docs/rfbproto.pdf,* RealVNC Ltd., July 2005.

[8] P. Ziewer, and H. Seidl, "Transparent Teleteaching", *19th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education (ASCILITE)*, Auckland, New Zealand, Dec. 2002, (2), pp. 749-758.

[9] P. Ziewer, and H. Seidl, "Annotiertes Lecture Recording", *2. e-Learning Fachtagung Informatik (DeLFI'04)*, Gesellschaft für Informatik, Paderborn, Germany, Sep. 2004, pp. 43-54.

[10] P. Ziewer, "Navigational Indices and Full Text Search by Automated Analyses of Screen Recorded Data", *World Conference on E-Learning (E-Learn'04)*, Washington, D.C., Dec. 2004.