# Navigational Indices and Full-Text Search by Automated Analyses of Screen Recorded Data

Peter Ziewer
Department of Computer Science
Technical University of Munich, Germany
ziewer@in.tum.de

**Abstract:** Lecture recording provides learning material for local and distance education. Screen recording offers a flexible technique for lecture recording as it allows to capture virtually any material displayed during a presentation. Unfortunately, the navigational features and search abilities are rather limited, because grabbing and storing pixel data deprives the structure and content of the source documents. This paper presents how analyses of such pixel based recordings deliver useful navigational indices and provide full-text search automatically. In order to establish these research results, we have (partly) implemented the ideas within our own screen recording environment called TeleTeachingTool.

## Introduction

The wide-spread availability of powerful multimedia enhanced computer systems and improved network connections let computer aided teaching become more and more popular. This is the case for distance education as well as for local courses. There are various different scenarios of e-learning/teleteaching environments containing for example internet based training modules, video conferencing, internet discussion groups or learning platforms to manage student admissions. This article concentrates on *lecture recording*, which is conserving presentations for later playback. Lecture recording offers a rather cheap and fast possibility of content creation and therefore is called *lightweight* content creation (Kandzia et al. 2004). Such recordings deliver additional learning material. They are useful, because they conserve presentations and teachers' original wording. The most flexible recording technique, screen recording, offers rather restricted navigational and retrieval features. This paper will present how to loosen these restrictions and, furthermore, how to do so automatically (in order not to collide with the lightweight aspect).

We will start with a comparison of two conflictive recording techniques, one flexible in recording and the other structured in its output. The next chapter offers an introduction into the recording environment used as basis for this research. The following two chapters present how to acquire the data needed to provide navigation and retrieval features by automated analyses of unstructured screen recordings.

## Recording Styles: Screen Recording vs. Symbolic Representation

Lecture recording applications can be divided into two groups, which differ in the way of storing a presentation. One is *screen recording* and the other records *symbolic representation*. Screen recorders store the "output" of a presentation. A recorder digitally grabs pixel data from the presentation machine's desktop and thus allows lossless high quality recordings. The storage of pixel data is very flexible. It allows to conserve everything happening upon the presenter's screen, including arbitrary applications, pointer movements, animations and annotations (e.g. notes and sketches using an electronic pen). *Camtasia*[1] or *TeleTeachingTool*[2] are such screen recorders. Unfortunately, as screen recording means storing everything as pixel data, it does not preserve any document structure or textual content. Thus deprives us of comfortable navigation within recorded documents, which, according to (Lauer & Ottmann 2002), should be a major goal of lecture recording software. Ordinary screen recording does only offer timeline navigation. Furthermore, post-processing and retrieval possibilities are rather limited.

[1] Camtasia product site: http://www.techsmith.com/products/studio/default.asp
[2] TeleTeachingTool project site: http/TeleTeaching.uni-trier.de

Instead of storing pixel data, symbolic representation formats, such as PDF or PowerPoint, store text as sequence of characters, font and color. The format is interpreted to generate a visible (or printable) output. The interpretation can be modified. The font size can be adapted to fit the resolution of the viewer's machine. Recorders basically store presentations' sources ("input") enhanced with timestamps for timed playback and synchronization. As such recordings preserve document structures, they offer good navigational, post-processing and retrieval options. The structure allows to distinguish slides and thus enables slide-based navigation. The textual content can be searched and edited. This allows information retrieval, correction of scribal errors or insertion of additional explanations. Furthermore, symbolically represented recordings normally result in smaller file sizes. However, this technique is not as flexible as screen recording concerning the input formats and applications to be recorded. Presenters are bound to a special presentation software (and maybe a single operating system). *Authoring on the Fly*[1] (AOF) uses a proprietary format, which demands slides to be created with special applications (AOFwb or mlb) only. Import filters (e.g. for PowerPoint) improve the usability, but cannot compensate all incompatibilities. Also, such recording systems are typically designed to store the outputs of the presentation software only. The parallel use of multiple software (e.g. presenter and browser) during a presentation is rarely supported or not possible at all. This may force the presenter to change his/her teaching style concerning content creation and presentation process. Reuse of formerly created, but now incompatible slides, is impossible or needs at least a transformation process.

So we have flexible recording versus structured recordings. One way out of this dilemma is offered by *Lecturnity*[2], a commercial descendant of AOF. This recorder stores symbolical representation, but offers an integrated screen grabbing utility to capture external applications on demand. But this approach still has limited navigational and retrieval options for the screen recorded parts and is meant for presenters rarely using additional applications.

Our research followed an alternative approach, which is to improve navigation functionality and integrate retrieval options into the screen recording technique. In the next chapters I will present how to compensate the missing structure and textual contents of screen recordings by extracting metadata and generating navigational indices.

## Lecture Recording

### Screen Recording based on Virtual Network Computing

In order to conserve a lecture, a screen recorder needs to grab the graphical output. One possibility is screen grabbing, which periodically stores the complete desktop as bitmap. This results in a huge amount of data. Lowering the frame rate (bitmaps per second) reduces the amount of data, but also reduces the quality. Instead of a smooth playback, pointer movements and animations look jerky. Using standard video codecs for compression "does in most cases neither lead to acceptable quality nor to a reasonable size of the compressed file" (Lauer & Ottmann 2002). The *TechSmith Screen Capture Codec* used by the commercial screen recorder *Camtasia* provides lossless compression of screen content, but is only available for MS Windows systems.

Another approach of conserving the graphical output is offered by *Virtual Network Computing*[3] (VNC) (Richardson et al. 1998), which is a remote display system, that allows to view and control a computing desktop environment 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 (Richardson 2003). Unlike other remote display protocols, such as the X Window system, the RFB protocol is totally independent of operating/windowing systems and applications. It works at the framebuffer level. The display side of the protocol is based on a single graphics primitive: Put a rectangle of pixel data at a given x,y position. A framebuffer update represents a change from one valid framebuffer state to another. Instead of using a fixed frame rate, the RFB protocol is demand-driven by the client. That is, an update is only sent by the server in response to an explicit request from the client and, additionally, is only sent if the framebuffer has changed. This reduces network traffic and, in case of recording, file sizes. The input side of the RFB protocol is based on a standard workstation model of a keyboard and a multibutton pointing device. Whenever the user presses a key or button or moves the pointing device, the VNC client sends input events, which are passed to applications running at a VNC server. The server transmits any change of its framebuffer in

[1] Authoring on the Fly project site: http://ad.informatik.uni-freiburg.de/aof/
[2] Lecturnity Suite product site: http://www.lecturnity.de
[3] Virtual Network Computing project site: http://www.realvnc.com

return. VNC offers lossless compression of framebuffers and thus provides high quality screen recording. The provided framebuffer can be used as source for a screen grabber or timestamped RFB protocol messages can be recorded for later playback (Li et al. 1999, 2000). The timestamps are used to synchronize playback.

In my own implementation of a VNC based recorder, I experienced that logging the length of each recorded message is valuable. Although the RFB protocol uses messages, its communication is done via streams. Unfortunately the only way to determine the length of each message is to parse the input stream until the end of a message is reached. Buffering messages during recording allows to determine and store their lengths and thus facilitate reading or skipping of complete messages during post-processing and playback.

### The TeleTeachingTool Environment

Our chair's e-learning research started in 2001 with the project *Universitärer Lehrverbund Informatik*[1], a cooperative project of German universities. The project's aim was a (partial) virtualization of the Computer Science study. Our task was to broadcast and record our lectures. As we did not find software that met our requirements, we developed our own environment, called *TeleTeachingTool* (TTT). The TTT is a cross-platform broadcasting and recording environment for presentations offering VNC-based screen recording (enhanced with audio and video), storing timestamped RFB messages. It allows the parallel use of arbitrary applications, including teacher's choice presentation software. Recording is done in a transparent way, without influencing the presenter, as described in (Ziewer & Seidl 2002). The main aspect of transparent screen recording is to allow teachers to use a presentation computer in the way they would do without recording. Only a VNC server is needed to export the desktop. The recorder captures the VNC output. This provides us with a flexible and powerful high quality recording mechanism, allowing an easy and fast creation of huge databases of lecture recordings. Additionally, we have integrated simple, but effective, annotation methods: Underlining, highlighting and freehand drawing. All these annotations are not bound to the presentation software, but are applied to the desktop as a whole. Thus makes them usable for all applications. Although TTT is a screen recorder, we achieved better navigational features than only timeline navigation and we have integrated full-text search. The software and more than 200 hours of recordings of several Computer Science and Media Science courses are freely available at our project site: http/TeleTeaching.uni-trier.de.

## Navigation within Recordings

### Navigating by Indices

Lectures and presentations are sequential and so are their video style recordings. Students, who have not attended local lectures (e.g. distance students), may watch recordings in full length. But in general sequential playback is not wanted. Students rather like to locate and study parts of special interest and skip other sequences. However, as standard screen recording does not conserve the presentation/document structure, only sequential playback or at most navigation by time is possible. This makes navigating rather annoying, particularly if accessing takes a couple of seconds caused by buffering techniques (e.g. RealPlayer). Even if immediate random access is supported, searching is not as comfortable as wanted. Dragging a slider back and forth until a certain position is found, is time consuming and annoying. To improve navigation, indices are needed as navigation marks. Any timestamp and thus any recorded data associated with a timestamp is a possible index. However, mentionable indices are only those addressing points of interest, like switching to another slide, the beginning of a new chapter or an animation.

(Li et al. 2000) divide four broad classes of indices (originally specified in (Minneman et al. 1995)). They introduced a VNC based screen recording system, that not only stores timestamped framebuffer update messages, but also records user event messages (e.g. key events), which also can be used as indexing points.

Firstly, there are *intentional annotations*, which are indices the teacher creates during presentation especially for the purpose of indexing (to mark particular points of interest). Highlighting a text string on the screen leads VNC to automatically create a server cut text message, which can be used as index. A teacher has to remember to create such indices during presentation, but (s)he may forget in the heat of the moment. Additionally the Li et al. system allows

[1] Universitärer Lehrverbund Informatik project site: http://www.uli-campus.de

to insert brief notes which are stored as client cut text message to fit into the RFB protocol. If used during lecture, this feature may interrupt and distract the flaw of the presentation. The explicit insertion of brief notes is better suited for post-processing and, in this case, should be filed under *post-hoc indices* (see below). Intentional indices can be placed automatically. *TeleTask*[1] uses a PowerPoint plug-in to generate an index each time the teacher switches to another slide. That allows slide-based navigation. However, this feature demands the use of PowerPoint and, if the teacher forgets to start the plug-in, no such indices are recorded.

The second class are *side-effect indices*, which are activities whose primary purpose is not indexing, but provide indices, because these activities are automatically recorded. Examples are key events and pointer events, which the teacher generates in order to interact with the computer. If special keys are used for special purpose, e.g. page down to switch slides, good indices can be obtained. On the other hand does each pointer movement generate dozens of needless indices. A sequence of key events results in a text. Since at most only small texts are entered during a presentation, the benefit for later text search is rather limited. In case of entering passwords, recording of key events is even safety critical, as it allows any client to acquire teachers' passwords used during presentations. An alternative is to record only predefined events, like the key to switch slides or the usage of a mouse button. This leads to similar problems like *intentional annotations*, as keys are application dependent and must be defined prior to recording.

Another class of indices are *post-hoc indices*. They are manually created in a post editing process to mark points of interest. If placed well considered, for example to mark the beginning of a new chapter or an definition, they are very useful. Unfortunately such post-processing can hardly be automated, which makes it time consuming and thus undesirable for a (mostly) automatic lecture recording process. However, they may be a good way to allow students to add their own notes to recorded presentations, which is also a post editing process, but not a mandatory one.

Finally there are *derived indices*, which are produced by automated analysis of recorded data. Post-processing allows tweaking of parameters to achieve best results. New or improved index analyses can be applied to all recordings and so even very old lectures benefit from research enhancements. Additionally, as the indices are calculated after the presentation is finished, the creation cannot be forgotten during recording process. Well suited automated analyses provide useful *derived indices* in short time as I will show in the next paragraphs.

**Slides Detection**

*Navigation by slide* is one main feature a playback engine should offer (Lauer & Ottmann 2002). A slide is (mainly) an image shown to the audience. Screen recording stores these slide-images as pixel data. Switching slides during a presentation results in huge screen changes. In our VNC environment any screen change results in a framebuffer update. Therefore the timestamp of any recorded huge framebuffer update is a possible index for a slide change. "Huge" can be seen as *huge-by-byte* (length of message) or *huge-by-area* (area covered by framebuffer update).

As the TeleTeachingTool stores VNC messages with a corresponding timestamp and message length, computing indices in terms of huge-by-byte is straight forward. A suitable threshold value should be as low as possible to distinguish slides from smaller partial updates. The computer science course "Informatik I" of Prof. Seidl, recorded 2001/2002 in Trier was used for empirical studies. During the 28 lectures nearly 1000 slides were used and with the threshold set to 10000 bytes, more than 95 percent of them can be detected. In most cases the undetected slides consist of a few words or small sketches only. Such slides contain mostly background colored pixels and therefore achieves good compression ratio and results in framebuffer updates of less than 10000 bytes. However, a lower threshold value produces to many faulty indices. These are caused by periodic non-incremental framebuffer updates, which are used to calculate key frames for fast random access. On the other hand, we noticed that sometimes slide detection generates multiple indices for the same slide. However, these are no faulty detections, but a result of the presentation style. The teacher has annotated his slides using the built-in drawing features of the presentation software. Deleting annotations caused a redrawing of the screen (creating framebuffer update messages), which is faulty detected as a slide change. But these indices give at least a meaningful partition of a slide. Applied to the course "Abstract Machines for Compilers", presented by Prof. Seidl and Prof. Wilhelm, the slide detection algorithm showed similar results.

[1] Tele-Task product site: http://www.tele-task.de/

However, we encountered problems using the same algorithm and threshold for other lectures. This was especially the course "Grundlagen der Medienwissenschaft" of Prof. Bucher, recorded in Trier in 2002. That Media Science course was more complex in its graphical representation. The extensive use of large high colored images and figures leads to higher sizes in bytes. Not only was the formerly determined threshold not suitable, it was even not possible to determine an appropriate value at all. We have tried to calculate a threshold in relation to a given recording considering peak values. However, this was only sufficient for homogeneous recordings, where all slides were of similar size in bytes, which was not the case for that Media Science course. Its presentations mixed slides containing only simple text with others using huge colorful illustrations and thus brought our approach to its limits. Slide detection on a size-by-bytes basis for such presentations is not sufficient, because a fragment of a colorful illustration can be larger than a complete slide containing only text. This is even more severe mixing incremental and non-incremental updates like the TTT does. Unfortunately the RFB protocol offers no possibility to distinguish incremental from non-incremental updates. But as the TTT only requests partial non-incremental framebuffer updates, the covered area of such an update is less than five percent of the desktop resolution. Switching fullscreen slides, in contrast, results in a framebuffer update covering (nearly) the complete desktop.

The approach of determining slide switches using the covered area of framebuffer updates is more promising. However, the realization is more complicated. One framebuffer update message may contain several rectangles. To compute the covered area of an update, all headers of all of its rectangles must be read. As this is only possible by totally parsing each message, creating slide indices is a complex job. Therefore the recording format was changed to simplify slide detection by area. Framebuffer update messages are no longer a composition of rectangles, but consist only of one single rectangle. In combination with the length stored with each message, a faster access to rectangle headers, and thus the parameters to calculate the covered area, can be achieved. Rectangles belonging together are identified by their identical timestamps. Calculating the covered areas (for any used timestamp) and comparing them with an empirically acquired threshold, delivers the indices for possible slide changes. This works even for non homogeneous presentations, because the area covered by an update is independent of content and compression.

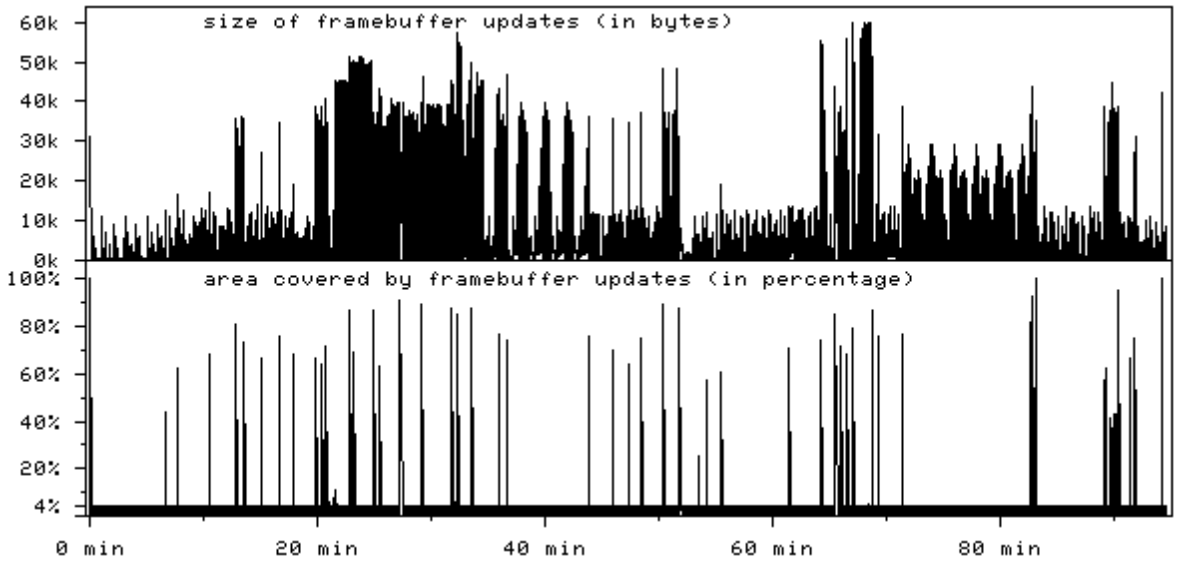


**Figure 1:** Framebuffer updates (size-by-byte vs. size-by-area)

(Figure 1) shows a comparison of the two slide detection algorithms for a non homogeneous presentation. For each framebuffer update its size in bytes and the covered area is shown. While peaks for the covered areas are clearly distinguishable, it is hard to find an appropriate threshold for the sizes in bytes. Between minute 72 and 82 you cannot see any peak among the covered areas, but there are many values up to 30k in the byte measurement graph. The byte values seem to repeat periodically, which is actually the fact. This is caused by the non-incremental updates used to calculate key frames. In this graph even the period of two minutes is identifiable. As these non-incremental updates are stripes containing 1/24 of a full image, they cause a line approximately at 4%, but no peaks, in the covered area graph. The presentation showed a scanned newspaper, which was very graphic intensive.

Slide detection results are good, but not perfect. Slides with overlays may generate too small updates, even in a by-area sense, and therefore do not generate an index. However, overlays are parts of a slide and finding at least the first overlay (which causes a huge update) means finding the beginning of the slide.

A suitable presentation of results has a fundamental impact to the usability of the playback software. Besides the conventional timeline navigation, the TTT offers a graphical overview of automatically computed slide indices. Clicking on a small preview image (thumbnail) causes a instantaneous playback of the corresponding slide. Additionally, accessing of the respectively previous and next slide is supported via buttons.

**Error Correction / Animation Detection**

A prediction about the quality of indices can be obtained by analysing a set of indices. Indices, that are followed by another one after short time, may not be that important. Analysis of our recordings showed various cases of such behaviour. Teachers searched a certain slide or skipped slides due to lack of time. They switched slides accidentally by pressing a key and then got back to the original slide immediately. This caused slides to be shown for a short period of time, maybe some milliseconds, resulting in multiple indices located in a short time interval. However, the viewer is only interested in the last of these indices, which describe the point where the presentation continues.

Unfortunately, there are cases of sequential indices where the point of interest is not the last, but the first index. So are animations, which can result in multiple huge framebuffer updates, but should be accessible at the beginning. Analyzing not only the timeshifts between indices, but also the length of a sequence of indices with small timeshifts, allows to classify indices. The TTT distinguishes real animations with lots of updates from the unimportant slides described above, which result in only a view sequential updates.

Animation detection not only works for indices caused by huge updates. Timeshifts between any updates can be analyzed to find animations. However, it is unclear how to distinguish designed animations, used for teaching purpose, from that caused by pointer movements or dragging a window, which are the same in sense of pixel data.

**Indices by Annotation Events**

The TTT environment offers annotation features like freehand drawing and highlighting. All annotations are stored as symbolic representation (instead of pixel data) and thus are potential indices. Annotations generate side-effect indices and, like pointer events, many of them may be useless. However, the *remove all annotations messages* can be used to detect a partition of a slide. In general, deleting all annotations means that some explanation or sketch is finished and a new part begins. The TTT can be configured to automatically send a remove all event while pressing designed keys to switch slides. We can use this event to find slide switches and to distinguish slide changes from animations, which do not send such an event. But as this is an optional feature, we cannot assure that all teachers use it, which causes this option not to be reliable, but auxiliary.

As we have seen above, we do not want all kinds of animations to be detected (e.g. no pointer movements). The recording of built-in annotations allows the detection of a special class of useful animations. Creating indices for freehand drawing events may give us fast access to sketches, figures or comments made on-the-fly. In most cases these are important annotations for students, which are not included in a printed version of a script. Furthermore, as all symbolically represented data can be edited, correcting erroneous annotations or inserting additional comments (by teachers or students) is possible.

**Automatic Script Generation**

Many teachers offer a script containing presentation slides, but not all do and for older recordings such a script may no longer be available. Slide indices allow to generate a script for each recording by taking a screenshot image of each detected slide. If the teacher uses any annotations, maybe to enrich his/her presentation by adding comments or doing on-the-fly sketches, even those additional information can be included in the script. As those annotations are created during the presentation, they are not available at the starting index of a slide, but just before removing the

actual slide and switching to the next one. Therefore, computing and storing full images of the recorded presentation at the latest timestamp previous to the next slide index, results in a script with annotations. Using the remove all events (see above) such a script can even include intermediate annotations. The TTT can automatically generate a script in HTML format including a clickable thumbnail overview.

## Information Retrieval: Text Recognition & Full-Text Search

A major advantage of text-based electronic media is an easy to use full-text search. By specifying a search pattern, mostly a string, a search engine offers a list of identified matches or subsequent searches offer the respectively next occurrence. Lecture recordings storing symbolically represented data (including text) obviously can offer full-text search. For screen recording systems, which store pixel data only, a presentation's text has to be acquired somehow. This can be done by using the (textual) sources of the presentation. Additionally a logical interconnection with the recording is needed, but can hardly be created automatically. Furthermore, as screen recording allows arbitrary applications to be used during presentations, their may be various different sources. This restricts the practicability of this approach as it may be laborious and/or complicated

Our research offers a mostly automated solution. As we have seen above, the TTT automatically generates a script. Applying external optical character recognition software to such a script delivers the text needed for full-text search. The logical interconnection between the extracted text and a recording is automatically given. As the slide indices provide the partition of the script, the texts associated with script pages are also associated with slide indices. The search algorithm returns the indices of the slides where the corresponding text matches the search pattern. This is a by-slide search function. However, we can make it more fine-grained. What we need are adequate indices to divide the recording into smaller sections. Presentations with overlays generate more frequent framebuffer changes, all of which can be used as index not only for navigation, but for splitting sections. As discussed above, the remove all events also offer a more fine grained partition.

Indices provide the logical interconnection between text and the recorded desktop. Search results are slide indices and thus we get the beginning of the teacher's comments about a slide (matching the search pattern), but not the precise comments about the searched string. The built-in annotation system of the TTT offers a possibility of an interconnection with the audio recording. The highlighting feature is generally used for marking words or text parts, which the teacher comments in exactly that moment. And as the presenter only points out important elements, we may have localized comments about interesting words, which are more likely to be searched. Using timestamps of highlighting events and applying text recognition to highlighted areas results in text that may be interconnected with audio comments about it. Thus should provide more precise search results. Pointer stop positions may also be used to create such interconnections, since the pointer can be used to point to something. Although detection via pointer events or animation detection can be done, it is hard to identify which area the pointer is pointing to or if it is pointing to something important at all. Therefore results may be very error-prone.

*Optical Character Recognition* (OCR) is used to extract text from various sources. Traditional text recognition algorithms are designed for printed sources. The algorithms are optimized for scanned high resolution images. As scanning is typically an error-prone process, error correction techniques do a re-alignment to assure horizontal characters or try to compensate shades, spots or other visual errors. In comparison to such scans, screenshots of typically 1024x768 pixels offer only a low resolution, but a high quality with no errors and a perfectly horizontal alignment. This may impair text recognition results, but in a brute force approach of using sophisticated OCR software (*OmnipagePro11*[1] and *ABBYY FineReader OCR7.0*[2]) for our automatically created script we achieved mostly good results for ordinary text. Results for formulas and sketches where rather bad. However, this is not a severe drawback, as almost only keywords are specified as search pattern. Outputs were very similar for colored and grey scaled sources. Black and white input offers better contrast, which reduces the error rate. Unfortunately all characters written in light colors were reduced to white and therefore were no longer distinguishable from the background. Further research is needed to improve recognition. A more sophisticated input may help, for example generating black and white images, but with consideration of contrasts, to keep all characters visible.

[1] OmniPage product site: http://www.scansoft.com/omnipage
[2] FineReader product site: http://www.abbyy.com/finereader/

As lecture recording enables to build up large multimedia databases containing hours of recordings, there is an increasing need for techniques, which enable users to localize specific information. An adequate presentation of the results is needed. This should offer easy to use navigation and a rating system. The TTT delivers search results via thumbnail navigation. Only thumbnails of indices matching the search results are presented in this view. Additionally, we offer an online full-text search for our recordings. In future the TTT should be possible to search a set of recordings. This is easy to implement, because the texts, which form the search base, are contained in the headers of the recorded lectures. Wolfgang Hürst discusses problems and demands of retrieval from recorded lectures in (Hürst 2003, Hürst et al. 2000). His research is based upon the symbolic representation recording environment AOF, but most results are transferable. The paper also focuses on searching audio streams. Speech recognition generates an audio transcript, which again can be searched for specific information. This will allow a better logical interconnection between search results and audio streams and thus delivers more fine grained results.

## Conclusion

Screen recording offers a flexible technique for lecture recording as it allows to capture virtually any material displayed during a presentation. Automated analyses compensates some drawbacks caused by the missing structure or symbolic representation of content. The TeleTeachingTool is a VNC based screen recorder, which uses automated slide detection to generate useful navigational indices. Animations can be distinguished from skipped slides in order to remove faulty or less useful indices. Easy to use navigation is provided via a graphical overview of slide indices. An editor may allow to manually edit indices and annotation in future if needed. Automated script generation (with or without annotations), followed by Optical Character Recognition, extracts texts used during presentation. In future speech recognition may create an audio transcript of the teacher's comments. Both are searchable and thus allow full-text search. The retrieval options needed to find specific information within large multimedia databases are at their basics, but further research should offer more sophisticated information retrieval for (databases of) recorded lectures in near future. Also an appropriate presentation of the results is needed.

## References


Hürst, W. (2003). Suche in aufgezeichneten Vorträgen und Vorlesungen. *Die 1. e-Learning Fachtagung Informatik, DeLFI 2003*, Gesellschaft für Informatik, Munich, Germany, Sep. 2003, 27-36.

Hürst, W., Müller, R. & Mayer, C. (2000). Multimedia Information Retrieval from Recorded Presentations. *ACM SIGIR 2000*, Athens, Greece.

Kandzia, P.-T., Kraus, G. & Ottmann, T. (2004). Der Universitäre Lehrverbund Informatik – eine Bilanz. *Softwaretechnik-Trends 24:1*, Gesellschaft für Informatik, 54-61,

Lauer, T. & Ottmann, Th. (2002). Means and Methods in Automatic Courseware Production: Experience and Technical Challenges. *World Conference on E-Learning (E-Learn 2002)*, Montreal, Canada, Oct. 2002.

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

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

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

Richardson, T. (2003). The RFB protocol. Version 3.7. *RealVNC Ltd*. http://www.realvnc.com/docs/rfbproto.pdf

Richardson, T., Stafford-Fraser, Q., Wood, K. & Hopper, A. (1998). Virtual Network Computing. *IEEE Internet Computing*, 2 (1), 33-38.

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