Integrating Corrections into Digital Ink Playback

Richard Anderson, Devy Pranowo, Craig Prince, Fred Videon
Department of Computer Science and Engineering
University of Washington
Seattle, Washington, USA

ABSTRACT
In this paper, we describe preliminary work on an ink editing application that allows an instructor to correct mistakes to digital ink written during a presentation that is to be archived. These corrections are then seamlessly reintegrated into the digital archive so that when the presentation is replayed the corrected ink is displayed instead of the original incorrect ink. We base our results on a system we have developed and prototype the work flow from initial presentation, through correction, updating the archive and playback. We show that a simple mechanism for correction is effective and low effort for the instructor. A key technical challenge that is addressed is the substitution of strokes by matching of the original and corrected ink.

Categories and Subject Descriptors
H.5.3 [**Group and Organizational Interfaces**]: Asynchronous Interaction

General Terms: Algorithms, Design, Human Factors.

Keywords: Digital Ink, Lecture Playback, Workflow, Editing, Classroom Technology.

1. Ink Replay
Many instructors use lecture capture technology to create audio and video versions of their lectures [1][5][7][8]. These can be high fidelity archives used for course export, or can be artifacts created for students to use in reviewing the course material. A standard viewing format shows the lecture slide along with an instructor video. Various controls are available to support navigation and replay. In this paper, we focus on lectures which include digital ink, where the instructor is using a device such as a Tablet PC to write on the electronic slides. The digital ink then becomes another information stream along with the slides, audio, and video. In replay, the temporal aspects of the ink are important to show order of expression and to allow synchronization with the audio stream.

Writing is recognized as a very important component of pedagogical presentation [6][4] and is often used to inject spontaneous material into a presentation. The persistence of writing, where text is displayed to the class is important since it allows a longer period for comprehension than the spoken word, and also becomes part of the record of the class session. However, an obvious drawback of handwriting is that it is often messy and mistake prone. Writing on a tablet device while lecturing presents additional challenges beyond whiteboard writing, potentially leading to further degradation in writing quality. Poor writing leads to confusion for students and may be more of a problem for off-line viewing of lectures since an instructor is not available for clarifications.

MM'09, October 19–24, 2009, Beijing, China.

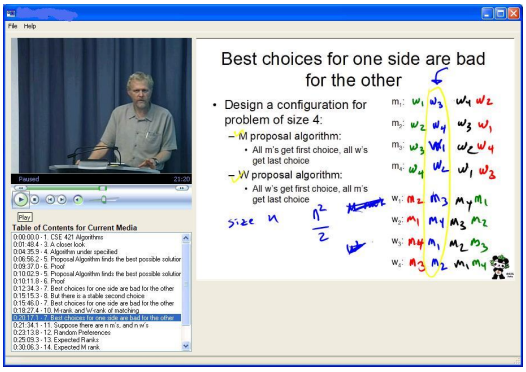

Figure 1. Replay tool showing writing on slides.

In this work, we investigate if we can develop an easy to use mechanism for an instructor to clean up and make corrections to digital ink for inclusion in the archived version of the lecture. The idea is that after lecture, the instructor will do a quick edit of the digital ink, and the revised ink will be used in the archive. The key for this to be successful is to make the editing very efficient for the instructor. Our proposed mechanism is for the instructor to make corrections directly on the static ink written on the lecture slides by writing over the ink to give a revised version. The system will then automatically identify the correspondence between the old ink and the corrected ink, and make the appropriate substitutions of ink into the archive. There are three main contributions of this work:

1. Creating a correction environment for the lecturer that is easy to use and efficient.
2. Developing mechanisms for accurately matching corrected ink strokes with the original ink.
3. Putting the pieces together into a complete tool chain from a presentation tool through an ink editor, archiver, and playback tool.

2. Ink Correction
The motivation for this work is the hypothesis that there would be a benefit to cleaning up digital ink prior to replay. Prior to considering such a system, it is reasonable to ask questions such as: How bad is instructor writing in practice? What types of errors are made in writing? When does the quality of writing compromise comprehensibility? To gather baseline data, we examined data from lectures at our university. We had access to a large collection of recorded lectures, as well as static images of digital ink on lecture

slides posted on the web. A range of instructors and sub disciplines of computer science were represented. The instructors' writing exhibited the expected range of errors and poor legibility. Common problems in textual ink included crossing out of characters, poorly defined symbols (especially subscripts or superscripts), and misspellings. Running out of space, resulting in compressed characters and non-horizontal text were relatively frequent. Diagrammatic ink would show additional problems, including poorly defined shapes and lines, and misalignment of components. Arrows in diagrams were particularly problematic. The figures below give examples of ink from real lecture slides along with the type of correction we would anticipate.

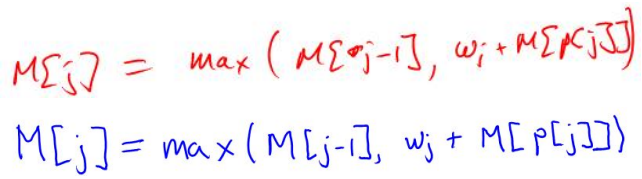

Figure 2. A formula showing crossing out and poorly defined subscripts and symbols.

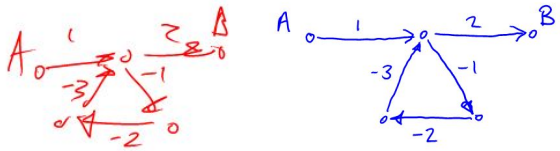

Figure 3. Diagrammatic ink with poorly drawn arrowheads and component.

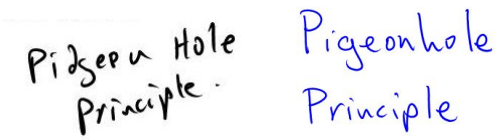

Figure 4. Writing at an angle with a misspelling.

Our idea for ink correction is that after lecture, the instructor will correct the static ink by writing corrected versions directly on top of the existing ink. In reviewing the types of corrections that would be needed on the slides, we observed that generally one would want to rewrite the entire line or diagram as opposed to making just single stroke corrections. We did a simple test to evaluate how much work it would be to correct ink from a lecture: we took the inked slides from several lectures and then traced the ink on the slides to simulate cleaning up the ink. For the samples we tried, which contained a moderate amount of writing, it took from 6 to 10 minutes to redo all of the ink from a 50 minute lecture. We did a second test (admittedly subjective), where we only corrected the ink that had "legibility problems", and noted that we were correcting between one quarter and one half the ink. These two tests suggest that ink correction need not be a burdensome post lecture task provided the tools are easy to use.

3. Correction UI

Our model for specifying ink correction is just to write the correction on the slide over the original ink. After a lecture is complete, the instructor will make a quick pass over his or her slides making corrections in the presentation tool, and then export the slides and ink to the archiver. One important aspect of the correction experience is that it is not necessary to know the order that the original ink was written when writing over the corrections. Especially with diagrammatic ink, it is very difficult to determine the order of writing from looking at the static diagram. The corrections are written without regard to the color of the ink. We assume that the stroke color will be inherited from the modified strokes.

There are other approaches that could be taken for ink correction. One would be to use ink recognition to attempt to automatically beautify the ink. However, the type of cleanup we want to do is well beyond the state of the art in ink recognition, so we did not pursue this approach. Another direction for editing the ink would be to allow explicit editing of the temporal sequence of strokes using a time line as is provided by many animation editors. This approach would give far better control than we are proposing. We have not taken this approach because our goal has been to make the editing as fast as possible, potentially compromising the quality. However, we anticipate that a production version of an ink editor would take a hybrid approach, and provide a timeline tool to allow more careful cleanup and fine tuning of results.

4. Stroke Matching

Stroke matching is used to determine the timing information for the new ink that is inserted into the archive. We find a correspondence between the new ink and the corrected ink, and then use the time stamps of old strokes on the corresponding new strokes. The specific problem that we need to solve is: given two sets of ink strokes, *Old* and *New*, representing the same drawing, find for each stroke *s* in *New* a corresponding stroke *s'* in *Old* and determine a start time for drawing *s* based on the start time of *s'*.

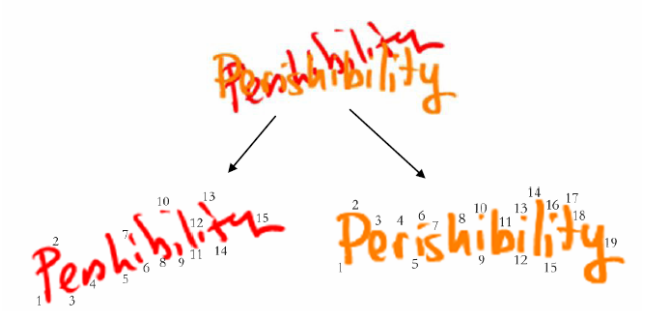

Figure 5. The stroke matching problem: the original writing is at an angle, and the corrected writing is horizontal. The problem is to match up the ink strokes in the corresponding versions.

The matching of strokes is not necessarily one-to-one. For example, the number of strokes used when writing a word may vary. In Figure 6, the 'd' in dog was written with a single stroke in S_A, but with two strokes in S_B.

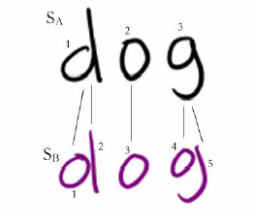

Figure 6. Correspondence of groups of strokes. Note that the 'd' in S_A is a single stroke, and the 'd' in S_B is two strokes. The correct matching is (1,1) (1,2) (2,3), (3,4), (3,5).

Instructors use a wide range of different types of inking while lecturing [6]. One classification is textual, diagrammatic, and attentional [4] where attentional ink consists of check marks and underlines to highlight current items of interest. Different algorithms are appropriate for different types of ink. Attentional ink is generally sparse, which makes the matching problem easy. For diagrammatic ink, we compute a similarity metric between ink strokes, and then use a closest correspondence. The current choice of similarity is based bounding boxes. For textual ink, we use a dynamic programming approach which we outline below. This depends upon first having the text segmented into lines, and then dynamic programming is used on matching the lines. A correct segmentation is important to getting decent results for the matching.

For matching ink from lines of text, we have two sets of strokes, s_1, …, s_n and t_1, … t_m. Each set of strokes is ordered in the temporal order of the writing. For each s_i and t_j, we can compute the goodness, G(i, j) of matching s_i with t_j. Our current goodness function is based on bounding boxes, although we have experimented with a range of functions. The optimal alignment if we only allow one-to-one matches is then computed using a standard dynamic programming algorithm with the objective function Opt[i,j] = max(Opt[i-1, j], Opt[i, j-1], Opt[i-1,j-1] + G(i,j)). To accommodate many-to-one matches, we define H(i,k,j) to be the goodness of matching s_i with $t_k \ldots t_j$, and we define K(k,i,j) to be the goodness of matching $s_k \ldots s_i$ with t_j. The new objective function is Opt[i,j] = max(Opt[i-1, j], Opt[i, j-1], $\max_k${Opt[i-1,k-1] + H(i,k,j)}, $\max_k${Opt[k-1, j-1] + K(k, i, j)}).

We expect that implementations of ink matching algorithms will produce errors (as do our current versions) when run on user's corrections. The most serious errors will probably arise from ink segmentation and classification. The question is what is the impact of these errors on the ink playback? If matching is incorrect, then ink strokes will appear out of order – either too early or too late. If the temporal gap is small, or the strokes are incidental the impact may not be great. This does suggest that taking additional temporal features into the matching may be important for achieving high quality.

5. SYSTEM

We developed a prototype system to evaluate our ink substitution ideas and demonstrate the process in an end to end fashion. In prior work, we have developed a classroom presentation system for the Tablet PC and tools for capturing lectures and playing back lectures. We extended these tools for our correction system. Our ink playback tool, which we will call LectureViewer packages audio, video, slides, and time stamped digital ink and allows replay through the custom application shown in Figure 1. The ink is represented with an xml format. For the prototype, we developed a standalone Tablet PC application that would read in LectureViewer files and allow editing of the ink. After corrections were completed, and time stamps for new ink assigned from the old ink, the ink stroke file for LectureViewer was updated. The new lecture could then be viewed with LectureViewer.

Figure 7 shows the application. A LectureViewer file is loaded, and individual slides can be navigated to and updated by writing new ink and erasing old ink. Two options are provided for invoking different matching algorithms.

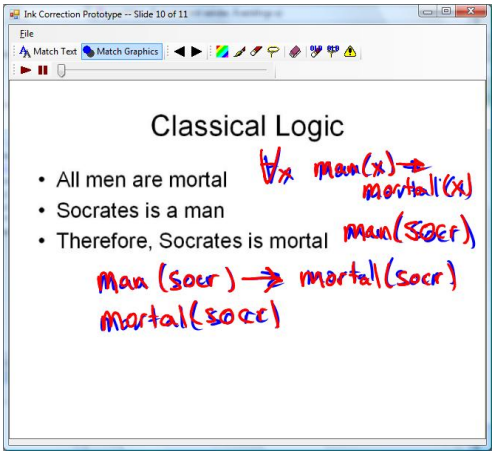

Figure 7. Ink correction prototype showing revised ink before matching.

6. EVALUATION

The basic evaluation was to take several full lectures with slides and ink, perform ink cleanup on all slides and watch the results. This provided validation of a number of aspects of the project:

- We were able to construct a system for ink correction that could handle full lectures.
- The corrections could be done quickly with a pen based application.
- The new static ink was more legible then the original and multiple mistakes were corrected per lecture.
- The dynamic ink in the corrected version appeared similar to the dynamic ink in the original. There was no visual evidence of the ink being "dubbed".

As a prototype there were some missing features, for example, not all of the features of the original presentation system could be handled. Ink which had been erased during the presentation could not be updated, since the application only worked with the final static images. The computational performance of the matching, which is done on a slide by slide basis needs to be sped up – slides with a substantial amount of writing took considerable processing time.

As described above, the correction process is to write new ink on top of the original ink. When the strokes are matched, any old strokes matched with new strokes are deleted. The unmatched old strokes are still available for inclusion in the archive – which allows some ink not to be corrected. The matching sometimes leaves some stray old strokes, which we deleted manually. The feedback on the unmatched strokes was important.

The most important part of the evaluation was to determine the impact of errors in the matching on the viewing experience. As a prototype, we expected a moderate number of errors in matching, and even with tuning of the implementations, errors in matching would happen (but hopefully at a lower frequency). There are three types of errors in matching – new strokes matched too early, new strokes matched too late, and unmatched old strokes that should have been matched. We manually removed the unmatched old strokes, so the issue was strokes matched at the wrong time. Strokes matched too early would appear as isolated characters and strokes matched too late would be missing. Figure 8 shows examples of

both of these. This turns out to be a significant problem because of the way writing is done on slides. It is common to spend a significant amount of time on a slide with episodes of writing and episodes of talking. This means that the incorrect state could be displayed for several minutes before the full writing appeared.

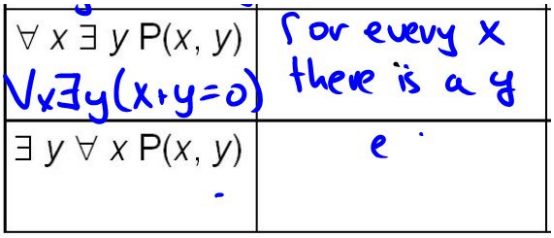

Figure 8. Strokes matched early and late: The cross strokes on the forall and the 'f' were matched too late, and the 'e' in the lower right was matched too early.

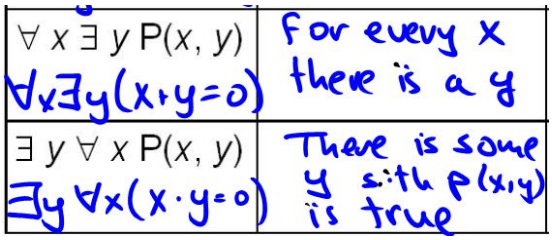

Figure 9. Completed ink with all strokes.

From our observations, we were convinced that the temporal mismatches are a serious problem if strokes are matched to the wrong episode of writing. We do not think that our initial approach of static matching is capable of avoiding all of these errors. Our plan is to augment our correction tool with a mechanism that allows corrections to be made one writing episode at a time, instead of all at once. This may increase the correction time slightly, but will give much better matching results.

Once case where our initial prototype behaves poorly is when diagrams are created with multiple phases of writing on top of each other. This is a common practice when using diagrams to explain concepts, where diagrams are updated in multiple phases [2]. The writing from the different phases overlaps, making it difficult or impossible to make corrections to all layers simultaneously. Fortunately, the solution of making phases available in sequence can be used to resolve this. It is common for diagrams to be drawn in three or four phases, although we have observed up to eight or ten. The phases can easily be identified by pauses in the writing.

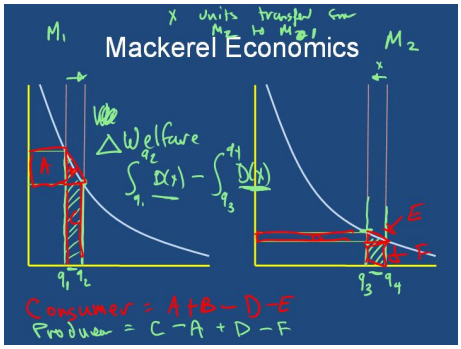

Figure 10. Diagrammatic ink drawn in multiple phases.

7. CONCLUSION AND FUTURE WORK

We have created an initial tool for allowing presenters (especially teachers) to edit the digital ink in lecture archives after giving a presentation. Our focus was on providing a lightweight way for corrections to annotations to be seamlessly re-integrated into a lecture as if they had originally been written during the lecture.

We consider our initial result to be positive. Our prototype allows corrections to be made, and can be applied to existing lecture archives, and we have verified that the effort in making the corrections is modest. There are a number of additional steps necessary to have this integrated into tools and to get instructors to take advantage of it. We believe that ease of use is paramount. Although instructors want to make high quality artifacts available to students, they will be discouraged if significant effort is required. On the replay side, it is very important to maintain temporal accuracy and avoid artifacts or missing ink which is visible for a long period of time.

8. REFERENCES

[1] Abowd, G. Classroom 2000: An Experiment with the Instrumentation of a Living Educational Environment. In IBM Systems Journal, 38(4), 1999, pp. 508-530.

[2] Anderson, R., Anderson, R., Hoyer, C., Prince, C., Su, J., Videon, F., and Wolfman, S., A Study of Diagrammatic Ink in Lecture. Computers and Graphics 29 (2005), pp. 480-489

[3] Anderson, R., Anderson, R., Hoyer, C., Prince, C. Su, J., Videon, F., and Wolfman, S., A Study of Digital Ink in Lecture Presentation, CHI 2004, pp. 567-574.

[4] Anderson, R., Hoyer, C., Prince, C., Su, J., Videon, F., and Wolfman, S., Speech, Ink, and Slides: The interaction of content channels, ACM Multimedia '04, pp. 796-803, 2004.

[5] Hurst, W., Indexing, searching and skimming of multimedia documents containing recorded lectures and live presentations, ACM Multimedia '03, 450-451, 2003.

[6] Lanir, R., Booth, K., and Findlater, L., Observing Presenters' Use of Visual Aids to Inform the Design of Classroom Presentation Software, CHI 2008, pp., 695-404.

[7] Lui, K-Y., and Chen, H-Y., Exploring Media Correlation and Synchronization for Navigated Hypermedia Documents. 13th ACM Multimedia Conference, 2005, pp. 61-70.

[8] Mukhopadhyay, S., and Smith, B., Passive Capture and Structuring of Lectures, ACM Multimedia '99, pp 477-487, 1999.