Comparing and Managing Multiple Versions of Slide Presentations

Steven M. Drucker *Georg Petschnigg*
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{sdrucker|georgp}@microsoft.com

Maneesh Agrawala
University of California, Berkeley
615 Soda Hall, Mail Code #1776
Berkeley, CA 94720-1776, USA
maneesh@cs.berkeley.edu

ABSTRACT

Despite the ubiquity of slide presentations, managing multiple presentations remains a challenge. Understanding how multiple versions of a presentation are related to one another, assembling new presentations from existing presentations, and collaborating to create and edit presentations are difficult tasks. In this paper, we explore techniques for comparing and managing multiple slide presentations. We propose a general comparison framework for computing similarities and differences between slides. Based on this framework we develop an interactive tool for visually comparing multiple presentations. The interactive visualization facilitates understanding how presentations have evolved over time. We show how the interactive tool can be used to assemble new presentations from a collection of older ones and to merge changes from multiple presentation authors.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General Terms: Algorithms, Design, Human Factors.

Keywords: Slide presentations, versions, distance metrics, correspondence, alignment

1 INTRODUCTION

Slide presentations have become a ubiquitous means of sharing information. In 2001, Microsoft estimated that at least 30 million PowerPoint presentations were created every day [19]. Knowledge workers often maintain collections of hundreds of presentations [3]. Moreover, it is common to create multiple versions of a presentation, adapting it as necessary to the audience or to other presentation constraints. One version may be designed as a 20 minute conference presentation for researchers, while another version may be designed as an hour long class for undergraduate students. Each version contains different aspects of the content.

A common approach to building a new presentation is to study the collection of older versions and then assemble together the appropriate pieces from the collection. Similarly, when collaborating with others on creating a presentation, the collaborators will often start from a common template, then separately fill in sections on their own and finally assemble the different versions together. Yet, current presentation creation tools [1, 12, 24] provide little support for working with multiple versions of a presentation simultaneously. The result is that assembling a new presentation from older versions can be very tedious.

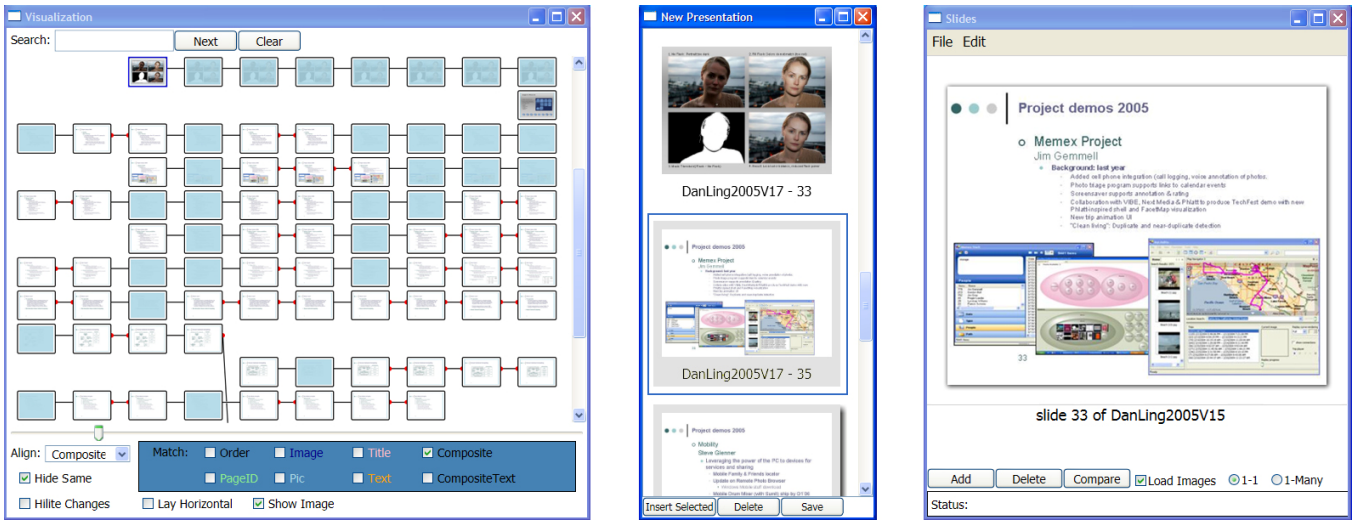
In this paper we present new techniques and tools for visually comparing and managing multiple versions of slide presentations. Our work makes three main contributions:

Comparison framework: We develop a framework for comparing presentations to identify the subsets of slides that are similar across each version. There are a number of ways to measure similarity between presentations, including pixel-level image differences between slides, differences between the text on each slide, etc. We propose several such distance measures and discuss how they reveal the underlying similarities and differences between presentations.

Interactive visualization: We provide an interactive tool for viewing multiple versions of a presentation. Users can examine differences between presentations along any of the distance measures computed by our comparison framework. The visualization is designed to help users understand how the presentation has evolved from version to version and determine when different portions crystallized into final form. Users can identify sections of the presentation that changed repeatedly. Such volatility might indicate problematic areas of the presentation and can help users understand the work that went into producing the presentation.

Interactive assembly: Our interactive tool also facilitates assembly of new presentations from the existing versions. Users can select subsets of slides from any version and copy them into a new presentation. The tight integration of visualization and assembly allows users to see the history of a presentation and combine relevant parts into the new presentation. Such an assembly tool is especially useful for collaborative production of presentations. Authors can independently edit the presentation and then use our assembly tool to decide which portions of each version to coalesce into the final presentation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UIST'06, October 15–18, 2006, Montreux, Switzerland.
Copyright 2006 ACM 1-59593-313-1/06/0010...$5.00.



Visual Comparison Window | Presentation Assembly Window | Slide Preview Window

Figure 1: Our interactive visualization and assembly tool is comprised of a Visual Comparison window (left), a Presentation Assembly window (middle) and a Slide Preview window (right). Users examine multiple presentations (each column of the Visual Comparison window shows a different presentation) and find the similarities and differences between them. Users can select any subset of slides from the Visual Comparison window and assemble them into a new presentation. The Slide Preview window allows users to inspect one slide and its alternate versions in greater detail.

A screenshot of our tool is shown in Figure 1. In this case, the Visual Comparison Window (Figure 1 left) shows 10 versions of a presentation – one per column. Lines linking slides and alignments between slides indicate slides that are similar to one another from one version to the next. We discuss our comparison framework in Section 3 and then show how it is used to generate the visualization of multiple presentations in Section 4. Users can select any subset of slides from the Visual Comparison Window and copy them into the Assembly Window (Figure 1 middle) to create a new presentation. A gray border in the Assembly window indicates that several slightly different versions of the slide are available. Users can also select a single slide either in the Visual Comparison Window or in the Assembly Window and an enlarged version of it appears in the Slide Preview window (Figure 1 right). The selected slide is highlighted with a blue border in both the Visual Comparison Window and the Assembly Window. We describe the interactive assembly process in Section 5. We provide examples showing how our system can be used to compare and manage multiple presentations in Section 6 and conclude in Section 7.

2 RELATED WORK

Finding the similarities and differences between two or more datasets is a problem that occurs in many contexts. File differencing tools such as UNIX's *diff* [10] highlight line level changes between two documents. These programs treat files as an ordered sequence of lines and typically compute the Longest Common Subsequence (LCS) of lines between two files using dynamic programming techniques [9, 11]. The LCS algorithm is related to the concept of string edit distance first introduced by Levenshtein [11]. The string edit distance is defined as the minimum number of operations (e.g. insertions, deletions, and substitutions) required to convert one string into another. File differencing programs based on edit-distance are often used by programmers to find all the lines of codes that were inserted, deleted or changed between two versions of a file. Similar techniques have been used to automatically detect plagiarism [18] and to find the best alignment between genetic sequences [15]. We use string edit distance to help compute distances between slides (Section 3.2.2), find corresponding slides between presentations (Section 3.3), and align slides in our visualization (Section 4.1).

Computing differences between data sets solves only part of the problem. For large data sets it is essential to provide visualizations that depict the changes and make it easy for viewers to focus on the similarities and differences between versions. SeeSoft [5] and SeeSys [6] provide focus+context tools for visualizing differences in text files. Viégas et al. [22] developed the *history flow* system to visually compare the changes made to Wikipedia articles. Using history flow they uncover a variety of patterns of cooperation and conflict that arose naturally as authors collectively created and edited Wikipedia. Their system is aimed at visualizing hundreds of versions of text documents. While we draw on this work for inspiration, our work is aimed at comparing slide presentations that contain graphics, images, and text. Unlike the earlier systems, we also provide tools for assembling new presentations from older versions.

Slide presentation tools such as PowerPoint [12], Keynote [1], and OpenOffice Impress [17] usually focus on providing tools for creating and presenting a sequence of slides. While PowerPoint does provide a "track change" mode for merging changes between two presentations, it forces a specific workflow. Users must process slides one at a time and accept or reject changes. PowerPoint does not provide any

way of seeing an overview of all the differences between multiple presentations at once. Our system provides this overview and allows users to work with multiple presentations simultaneously.

While current commercial slide creation software focuses on producing a single linear sequence of slides, several research systems support multiple paths though a presentation. Pad[20] and CounterPoint [7] are zoomable interfaces that allow spatially positioning slides on an infinite canvas and support hyperlinked navigation to any slide in the presentation. Zellweger [23] has developed a system for building multimedia documents embedded with multiple scripted paths. Nelson et al.'s [16] Pallete system is a tangible, paper-based interface for organizing presentations. More recently, Moscovich et al. [14] have developed a system that allows users to choose between multiple paths, on-the-fly, as they are giving the talk. All of these systems facilitate the process of customizing a presentation. Our system is aimed at comparing and managing multiple presentations, therefore, it is largely orthogonal to these techniques and could be used in conjunction with any of them.

3 COMPARISON FRAMEWORK

The goal of comparing two slide presentations is to identify all of the similarities and differences between the slides within each presentation. The key step is to find the "best" matching slide from the second presentation for each slide in the first presentation. We can compute such matching correspondences with respect to many different features of the slides. Moreover, the best correspondence with respect to one feature may not be the best with respect to another feature. Thus, we have developed a general framework for computing correspondences with respect to a variety of features. Users can easily extend the framework to compute new types of correspondence.

For each slide in a presentation, we extract a set of basic *features* (discussed in Section 3.1) and then use feature-specific *distance operators* (Section 3.2) to compute a set of distances between pairs of slides. Next we apply *correspondence operators* (Section 3.3) to find the "best" match between a slide in the first presentation and a slide in the second presentation.

3.1 Slide Features

We consider any basic descriptive element of a slide to be a feature. The graphical elements including vector drawings, images, charts and tables, as well as the text contained on a slide are all examples of slide features. We also consider the bitmap image of a slide to be a feature of it. Other examples of slide features include the position of text boxes and graphic elements, background graphics or colors, formatting parameters of text, header text, footer text, note text, and animation settings. Some features are specific to the tool used to create the presentation. For example, PowerPoint assigns a unique ID for each slide and for each image on a slide. For a comprehensive list of object model level features see the file format specifications of Microsoft

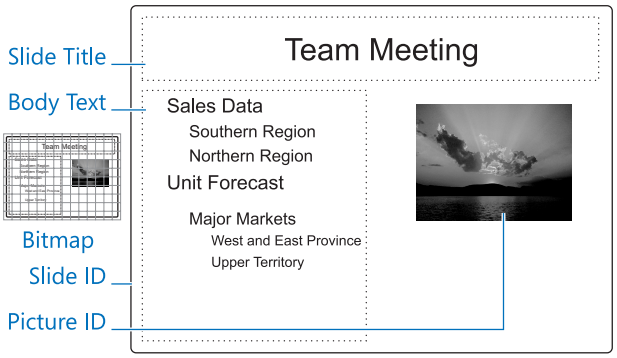


Figure 2: Slide features currently used in our comparison framework.

PowerPoint [13], Apple's Keynote [1] or OpenOffice's Impress [17].

Figure 2 describes the features that we use in our framework. Although our implementation currently includes only a few basic features that we have found most useful for comparing presentations, the framework could easily be extended to handle other descriptive features of a slide.

3.2 Distance Operators

The first step in comparing two presentations is to compute distances between the slides with respect to underlying slide features. Each distance operator takes two presentations and computes a distance between every pair of slides with the first slide from the first presentation, and the second slide from the second presentation. All of our current distance operators are symmetric, though the framework can handle asymmetric distance operators as well.

3.2.1 Image Distance

We compute the image distance between two slides by calculating the mean square error (MSE) between their bitmap images. The MSE measures visual similarity with smaller values indicating greater similarity. A MSE of zero means that the two slides are visually identical to one another, while a large MSE implies that there may be large visual differences between the slides.

A drawback of MSE, is that it often does not match human perceptions of visual differences. For example, slightly changing the position of an image between two slides can produce a large MSE, even though the slides will look very similar. Similarly, a minor insertion or deletion of text that causes the text to reflow will produce a relatively large MSE. Yet, the meaning of the text may not have changed at all. Alternate image distance measures based on sub-region comparisons may be less sensitive to small changes in slide layout. Image distance metrics based on models of human visual perception might also provide more meaningful distances. Nevertheless we have found MSE to be a very useful measure of slide similarity, especially for identifying visually identical slides.

3.2.2 Text Distance (Levenshtein or Edit Distance)

As mentioned previously, the string edit distance measures the minimum number of operations required to convert one

string into another string. Our text distance operator uses Levenshtein's dynamic programming algorithm [11] to efficiently compute the edit distance between textual features such as Slide Title and Body Text. The algorithm builds a matrix of costs required to convert one string into another and then reports the minimum cost path through this matrix. For completeness, we provide a brief description of the string edit distance algorithm in Appendix A.

Another approach for comparing text strings is based on a trigram model [21]. The idea is to build a histogram of all three letter sequences of characters within each string. The distance between the strings is then computed as the dot product of the histograms. The advantage of this approach is that it is less sensitive than string edit distance to rearrangements of text. For example, reordering bullet points in the body text of a slide will yield a large string edit distance but a relatively low trigram distance. In our system, we currently use string edit distance and have found that it gives a good measure of text similarity. We leave it as future work to compare the trigram approach with string edit distance for presentation comparisons.

3.2.3 Slide ID and Picture ID Distances

Slide IDs and Picture IDs are PowerPoint specific features. They are unique identifiers for each slide and each image on a slide. Once created, they remain fixed for the lifetime of a document. Thus, we can directly compare these IDs to identify matching slides and images between two versions of a presentation. The Slide ID distance operator returns 0 if the slide IDs match and a very large value when they do not match. The Picture ID distance operator determines the maximum number of images in common between two slides and returns the reciprocal of that number plus 1. Thus slides with many matches have lower distances than those slides with fewer or no matches. If there are zero Picture ID matches the operator returns a very large value.

While a Slide ID distance of 0 shows that two slides once started out as identical, there is no guarantee that the slides remain similar. The slides could have been heavily edited within each presentation independently. Similarly even if Slide IDs differ, the slides may be visually identical. The simple act of copy/pasting (as opposed to cut/paste) will produce identical slides with different Slide IDs. Nevertheless, the Slide ID distance does provide a measure of slide similarity that is insensitive to subsequent slide edits.

3.2.4 Composite Distances

Our system also supports composite distance operators that combine several basic distance operators into a single function. For instance we have found it useful to combine the image and text distances into a single composite distance. For each pair of slides we normalize the image and text distances so that they are roughly in the same range and can be compared meaningfully. We then use the minimum of the two normalized distances as our composite distance. This composite distance returns a single number that can be used to compare slides that contain extensive amounts of text and those that contain no text, but only images.

One challenge in developing such composite distance functions is normalizing the individual distance operators so that they can be meaningfully combined with one another. For example, image distances are measured in color space, while text distances are measured with respect to the number of insertions/deletions required to convert one string into another. In our current implementation we choose the normalization factors by manually looking at and adjusting the ranges of the individual distances.

A second challenge is to choose how to combine the normalized distances. Taking the minimum distance essentially considers only the best matching feature as the representative distance between slide pairs. Another approach is to compute a weighted sum of the individual distances. While the user could then control the importance of each distance operator by setting its weight, choosing appropriate weights may be a difficult task.

3.3 Slide Correspondence Operators

To find the best match between slides in each presentation we compute slide to slide *correspondences*. These correspondences are the key to identifying the changes between presentations. As we will show in Section 4 our interactive visualization tool is designed to visually depict these correspondences so that users can quickly see similarities and differences between multiple presentations.

Correspondence operators take two presentations and a distance operator as input and yield a mapping between each slide in the first presentation and its best matching slide in the second presentation. In our implementation, each slide can appear in at most one match, and if no good match is found the operator can leave a slide unmatched.

3.3.1 Minimum Distance Correspondence

A simple technique for computing correspondence is to match each slide in the first presentation with the minimum distance slide in the second presentation. While this approach could be used in conjunction with any of our distance operators, it has several drawbacks. If multiple slides are at the same minimum distance, it is unclear how to pick the best match from amongst them. There is also no provision for leaving a slide unmatched; even if none of the slides in the second presentation is a "good" match, this technique will still generate a correspondence.

3.3.2 String Edit Distance Based Correspondences

We can think of each presentation as a sequence of symbols and then compute correspondences using the string edit distance algorithm described in Section 3.2.2 and Appendix A. We assume that two slides match when the distance between them is less than a user-specified minimum threshold. Backtracking through the resultant cost matrix, we can recover a correspondence for each slide. Note that the string edit distance algorithm cannot determine if blocks of slides have moved from one position to another between presenta-

tions. It only reports slide insertions, deletions, and substitutions. Thus, we cannot use string alignment to find correspondences between slides that cross over other groups of corresponding slides, which is common in presentations.

3.3.3 Greedy-Thresholded Correspondence

Heckel [8] presents a greedy algorithm for computing correspondences between sequences of symbols. This approach finds uniquely corresponding symbols, removes them from the potential set for consideration, and then expands the search from those symbols to adjacent symbols in order to find the best correspondences. The algorithm iterates until no more matches are found.

Heckel's algorithm requires unique matches between symbols. Since we compute feature distances rather than unique matches we cannot directly apply Heckel's technique and instead adapt it as follows:

1. Given a distance operator sort the distances between all pairs of slides from least to greatest.
2. Create a correspondences between the minimum distance pair subject to a distance threshold ε.
3. Remove both slides from further consideration.
4. Continue from step 2 until no more correspondences can be found.

We introduce a minimum distance threshold ε in step 2 so that slides that are significantly different cannot be matched to one another. We have found that good values for ε depend on the type of distance operator being used. We use the following thresholds: image-based distance – 100 units of mean square error, string edit distance - 30 operations, slide and picture ID distances - only allow correspondence when all IDs match.

White this greedy algorithm has worked well on the examples we have tested, it can run into some problems. Like any greedy algorithm our approach may not always produce an optimal solution. In particular a slide in presentation 1 may not be matched to a minimum distance slide in presentation 2. In addition, our approach does not consider sequential proximity in computing correspondence. A slide at the beginning of presentation 1 may best match a slide near the end of presentation 2, but have a reasonably close match at the beginning of presentation 2. Our current algorithm would report the slide at the end of presentation 2. Heckel includes a notion of sequential proximity in his distance computation and we believe it is possible to extend our approach in a similar manner.

3.3.4 Composite Correspondences

Our correspondence operators can be computed with respect to any distance operator, including the composite distance operators. However, as we noted earlier it is not always clear how to normalize the individual distance operators to produce a meaningful composite distance. Therefore we have developed an alternative approach for combining multiple distance operators, but at the level of the correspondence operator.

Our approach is based on a voting scheme. We first compute correspondences using any set of distance and correspondence operators as described in sections 3.3.1-3.3.3. For a given slide in the first presentation each distance operator can generate a different minimum distance matching slide in the second presentation. We treat each minimum distance match as a vote for a particular correspondence and report the slide in the second presentation that receives the most votes as the corresponding slide. A tie in the voting means that there is disagreement between the distance operators on individual features. In such cases the slide in the first presentation is left unmatched. We have found that combining the image, text and Slide ID greedy-threshold correspondences using such a voting scheme is useful. The Slide ID correspondence essentially arbitrates between the image and text correspondences.

When changes affect many slides (such as a template change), image distances will be large between corresponding slides while other distances such as text, Slide ID and Picture ID distances may be small or identical. Our composite correspondence operators can detect template changes because they consider the variance between image distances and text, Slide ID, and Picture ID differences.

4 VISUALIZING MULTIPLE PRESENTATIONS

To help users understand similarities and differences in the presentations, we allow users to interactively generate visualizations that reveal correspondences between presentations. Examples of the types of visualizations we generate are shown in Figure 3. Each column represents a presentation and each rectangle within a column represents a slide. In the initial layout (Figure 3-a), the relative lengths of both presentations is immediately apparent.

4.1 Conveying Correspondence

Correspondence is conveyed through two visual representations. First, users can turn on lines that connect corresponding slides based on any of the distance and correspondence operators (Figure 3-b). The color of the line indicates the type of distance operator used (e.g., text distances, image distances). When users hover the cursor over a line, the numerical distance between the slides is shown.

Our second approach to visualizing correspondence is to align corresponding slides. We compute the minimum number of gaps required to maximize the number of corresponding pairs of slides that align between two presentations subject to the constraint that each presentation cannot modify the sequential ordering of the slides. (Figure 3-c). Note that as a result of this constraint, corresponding slides cannot always be aligned. For an example, the 6^{th} slide in the first presentation of Figure 3-c cannot be aligned with its corresponding slide, but a line can still be used to show that this slide corresponds to the 8^{th} slide in the second presentation.

We again use a string edit distance algorithm based on dynamic programming to compute slide alignment However, in this case, we use a modified version of Hirschberg's [9]

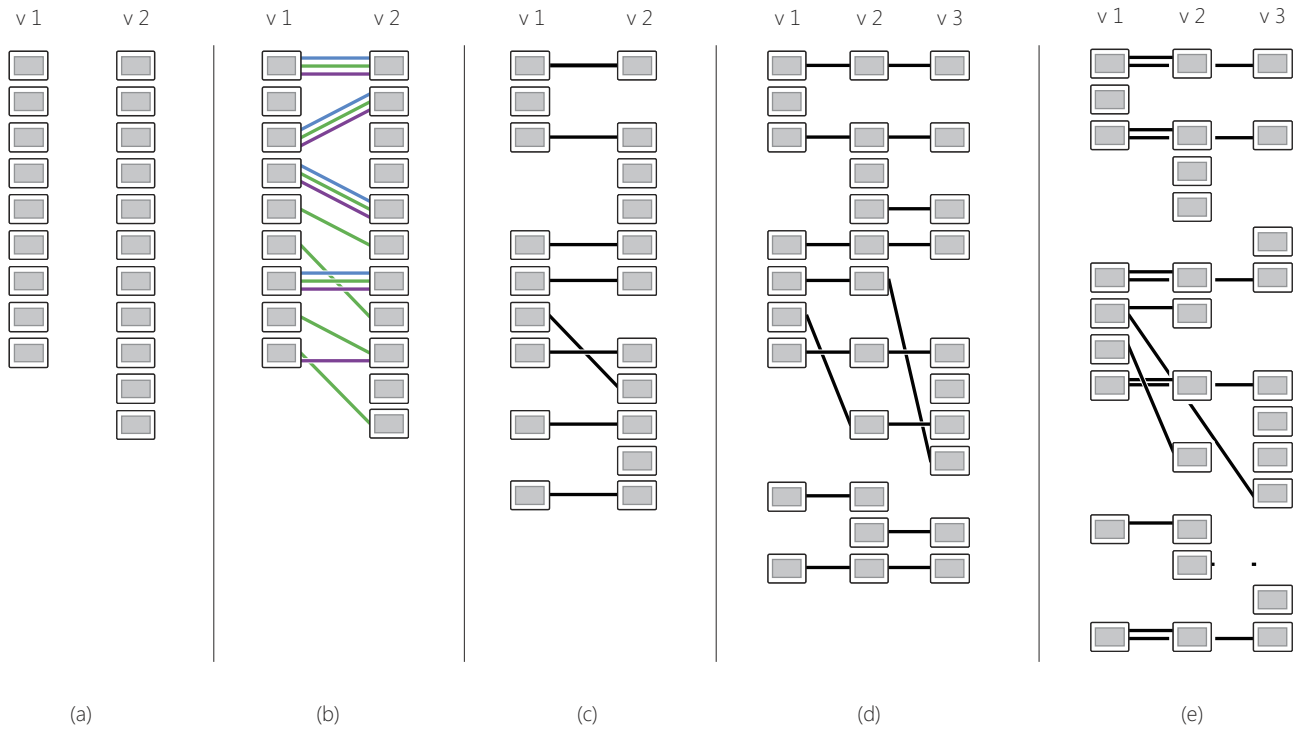


Figure 3: (a) Two presentations arranged in columns. (b) Lines connect corresponding slides. The color of the line indicates the type of distance operator used. For example, blue indicates image distance. (c) Presentations aligned using the Hirschberg [9] string matching algorithm. Alignment is based on correspondences computed using one type of distance operator, the lines depict correspondences using the same distance operator. (d) Multiple sequences compared serially – v1 compared with v2 and v2 compared with v3. (e) An alternate layout comparing one to many presentations, lines are drawn between slides in the first presentation and corresponding slides in the subsequent versions.

algorithm because it is more space-efficient than the more standard Levenshtein string matching algorithm. As more presentations are added to the comparison, gaps are adjusted throughout all the presentations to keep corresponding slides aligned when possible (see Figure 4).

We've also found it useful to highlight corresponding pairs of slides that are visually identical (i.e. with an image distance of 0). Visually identical corresponding slides can be dimmed to a light blue color. In addition, corresponding slides that are not visually identical can be linked using lines with red-colored end-caps. Both of these approaches help draw the user's attention to slides which have changed from version to version. The dimming is shown in Figures 1 and 6, while the end-caps are shown in Figures 1, 5, and 7.

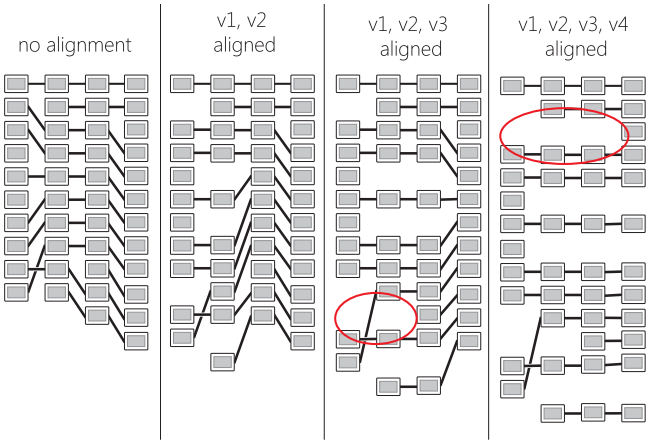


Figure 4: Alignment of multiple presentations: Gaps are inserted in both presentations 1 and 2 to achieve maximal alignment. As subsequent presentations are aligned, gaps must be inserted in all previous presentations to keep them all aligned.

4.2 Presentation to Presentation Visualizations

Our system provides two modes for visually comparing multiple versions of a presentation. The *sequential one-to-one comparison* mode assumes that the versions were created in a particular order and compares version 1 with version 2, version 2 with version 3 and so on. This mode is useful for tracking changes in the presentation as it directly depicts the evolution of the presentation from version to version. The *one-to-many comparison* mode compares a single base presentation to several alternative versions of it. This mode is most appropriate for seeing how a master presentation was assembled from earlier versions, or for collaboratively combining presentations that were simultaneously edited by multiple collaborators. Figures 1, 3 (b-d), 4 (a-d) and 5 all show sequential one-to-one comparison, while Figures 4-e, 6 and 7 show one-to-many comparisons.

4.3 Interacting with the Visualization

The user can interact with the visualization by using a slider to zoom out to see an overview of the changes, or to zoom into a particular slide or region of slides. Clicking on a slide will select it and bring up a full resolution slide in a slide preview window. The user can use the arrow keys on the keyboard to move the selection forward or backward within a presentation, or move between corresponding slides across presentations. By quickly moving back and forth between corresponding slides, the user can easily see visual differences in the slides in the slide preview window.

Checkboxes allow different correspondence links to be turned on and off, and a pull down menu allows the presentations to be aligned along any of the correspondences. The user can also select a slide and find similar slides along any distance operator. Images of slides can be turned on or off to just focus on the overall structure of changes. Slides that do not change along a particular distance operator can be dimmed to a light blue to help highlight only the changes.

5 ASSEMBLING PRESENTATIONS

Besides allowing analysis of the relationships between multiple presentations, the visualization tools also facilitate the assembly of new presentations. Users can select a set of slides from any presentation in the Visual Comparison Window and paste them into the new presentation in the Presentation Assembly Window.

Our system provides a number of techniques for selecting slides in the Visual Comparison Window; all the slides within a presentation can be selected by clicking on the presentation title and all slides that contain a given string can be selected by searching for the term.

In the newly assembled presentations, slides maintain their correspondences to slides in the older presentations and users can easily choose between alternate slides using the arrow keys. Slides that have visually distinguishable correspondences are outlined in gray to indicate that alternates are available.

6 IMPLEMENTATION

The system was implemented using the Microsoft Office Primary Interop Assemblies to access the object model for PowerPoint and automate the extraction of all the features contained on the slides. The visualization was developed using the Windows Presentation Framework, and a variant of Python called IronPython that uses the Common Language Runtime (CLR) which facilitated rapid development and allowed for convenient loading of modules for visual comparison, textual comparison, and PowerPoint interaction. The code is not currently optimized and takes approximately 1 minute to extract features and compare two moderate sized presentations (30 slides) on a 2 GHz computer with 1 Gb of RAM. The features and the comparisons are saved in XML files so that once run, the comparison will only re-run if the source presentations are altered.

7 RESULTS

Our results are depicted in Figures 5 – 7.

Figure 5 shows a visualization of 10 different versions of a presentation prepared by multiple authors for an executive review. The visualization depicts 387 slides. Each version of the presentation is sequentially compared to the next which allows for an analysis of the presentation over time. In versions 3 and 8, several slides have been added as indicated by the large insertion gaps. Conversely from versions 5 to 6, a four slide section was removed to shorten the presentation. From versions 7 to 8 a slide that occurred later in the presentation is moved earlier. Similarly, the visualization allows viewers to rapidly see the changes throughout the evolution of the presentation.

Figure 6 shows a one-to-many comparison where several authors edited a single base presentation and the system was then used to identify and coalesce changes. The visualization shows where authors spot the same typo and how different authors might suggest alternate changes to the flow of the presentation.

Figure 7 shows our system being used to assemble a presentation. Here the user prepares for a mid year review by pulling slides from two talks given earlier in the year. Our visualization lets the user compare the two presentations (Figure 7-a) and choose the desired slides. For example the second slide in the assembly is from version 2, the fifth slide from version 1. The gaps indicate slides that only exist in one version. Once assembly is complete, the user can save out a new version of the presentation and make modifications such as updating the title slide. Figure 7-b uses our one-to-many correspondence to compare the newly assembled presentation to the sources. This view directly shows which source presentation each slide came from.

8 CONCLUSIONS

We have presented a framework and set of visualization tools for analyzing and simultaneously presenting multiple presentations. These tools can be used to assist in the creation of new presentations and support a variety of work strategies from tracking changes for individuals, merging multiple versions, or assembling new presentations. Our visualizations can also give sociologists the tools to detect patterns in multiple versions of a slide presentation or even among all the presentations owned by a user or organization.

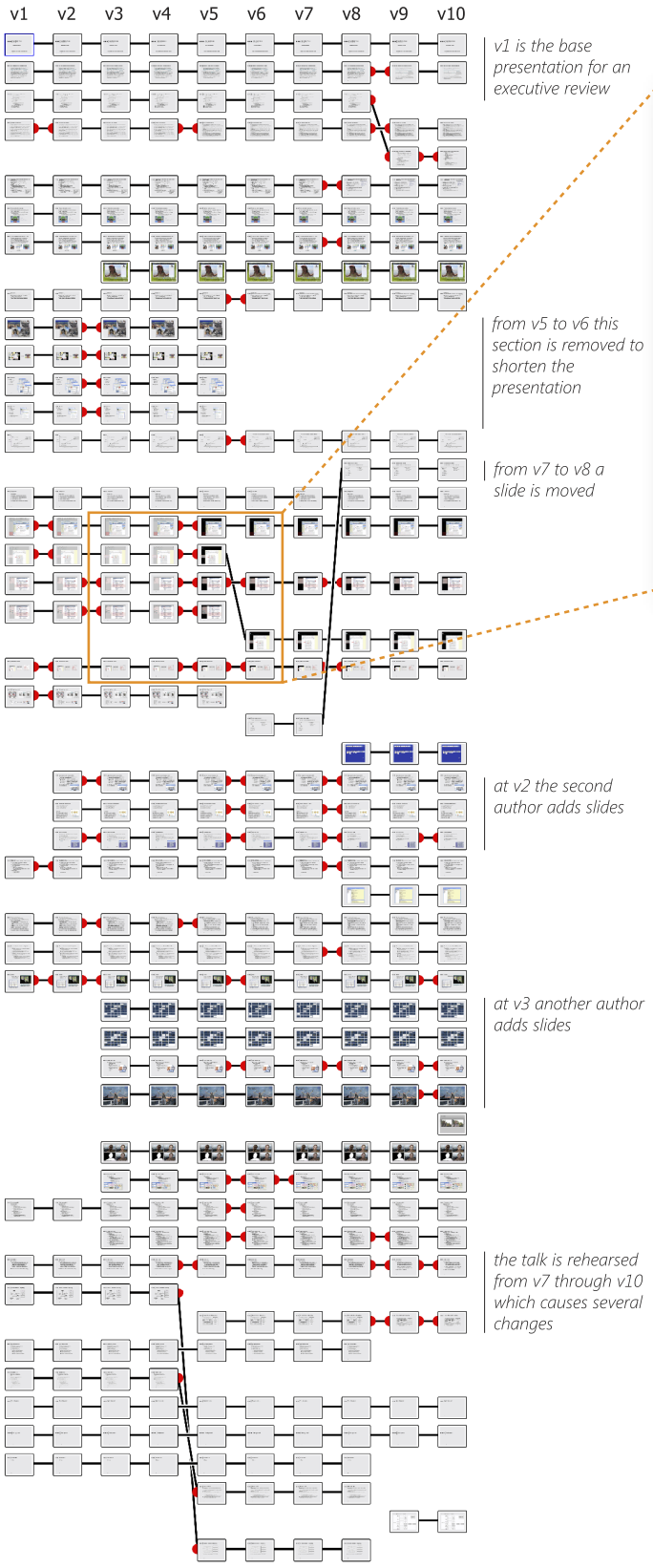


Figure 5: Ten versions of a presentation prepared for a review. The presentation consists of 387 slides. Gaps denote where sections where added or removed (e.g. between v5 and v6 a large section was removed to shorten the talk).

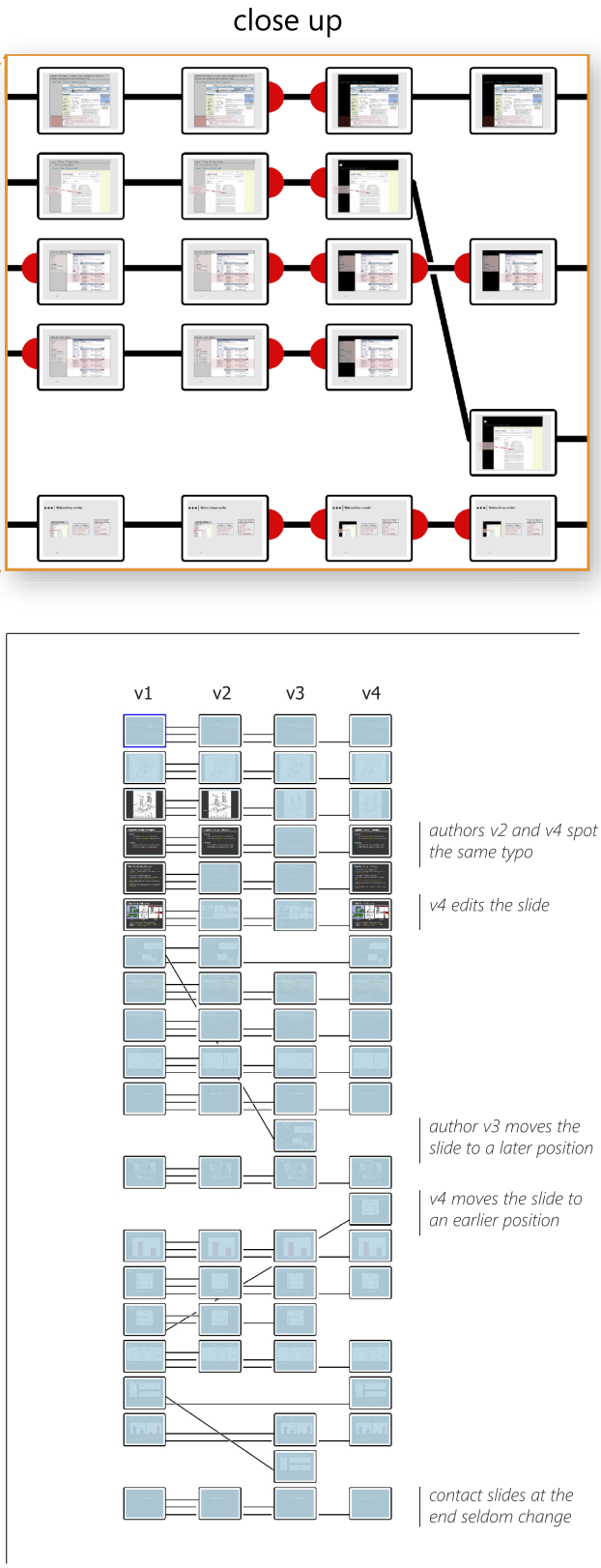
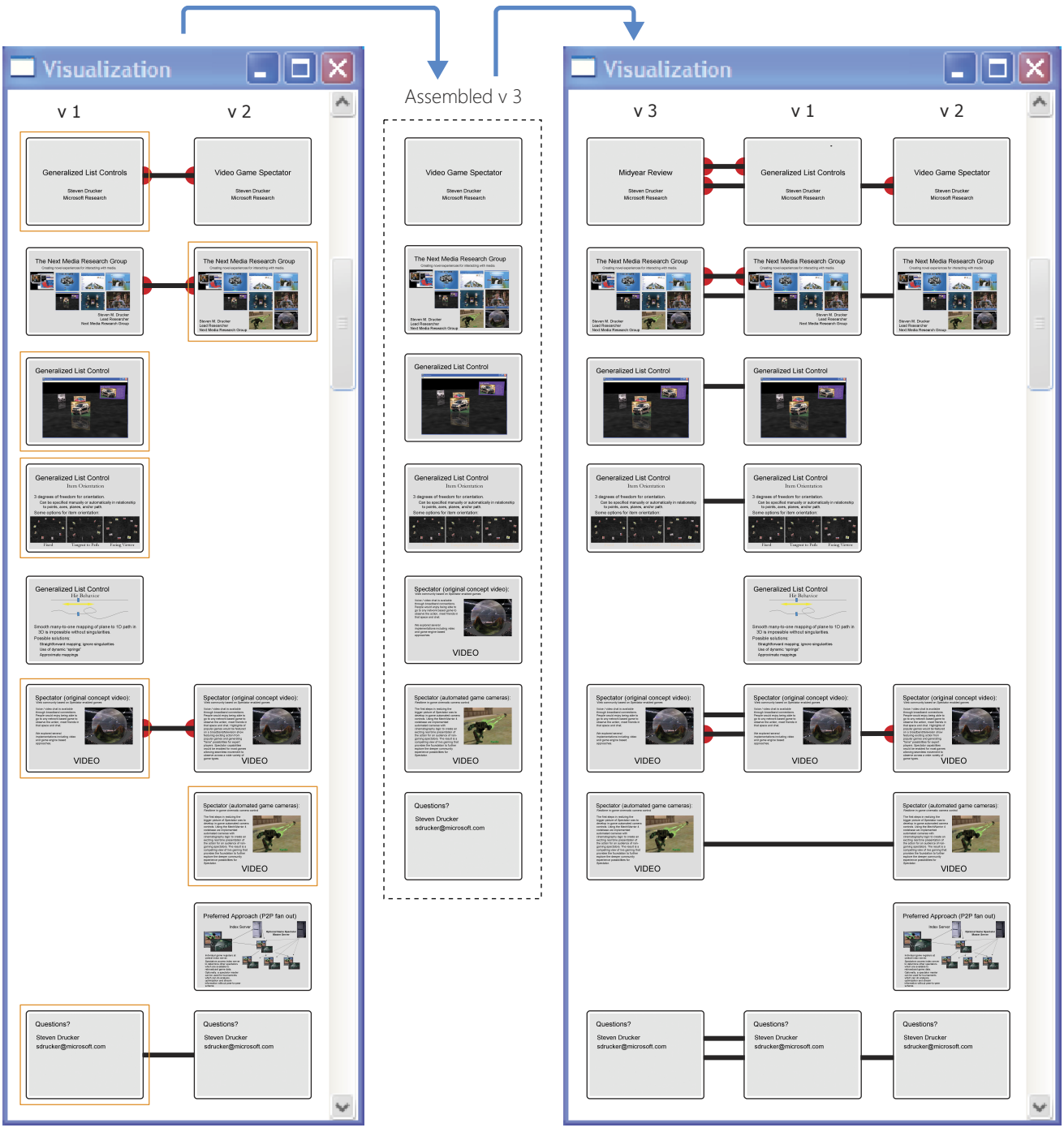


Figure 6: Merging changes using a one-to-many comparison of a base presentation v1 that has been edited by 3 different authors (v2, v3, v4).Slides that are visually identical are dimmed to light blue.



(a) (b)

Figure 7: (a) Using our system for presentation assembly. v1 and v2 are two related presentations. The sequential comparison makes it easy to choose slides from the two versions: Alternate versions of a slide are aligned, and slides that have changed under the image metric are denoted with red end-caps. The user can pick the desired slides (outlined in gray) and add them to the presentation assembly. In (b) the assembled presentation is compared to its source versions. Our one-to-many comparison shows the source presentation each slide in our new assembled presentation came from.

9 REFERENCES

1. Apple. Keynote. 2005. www.apple.com/keynote; Keynote Document Reference: developer.apple.com/technotes/tn2002/tn2073.html
2. Baker, M.J., and Eick, S. 1995. Space-filling software visualization. *Journal of Visual Language and Computing.* 6. pp.119-133
3. Brand, John. 2004. Presentation (Mis)management: Content and Collaboration Strategies, Delta 3057. Sept 14, 2004.
4. Charras, C. Lecroq, T. Sequence comparison. www-igm.univ-mlv.fr/~lecroq/seqcomp/index.html
5. Eick, S., Steffen, J.L., and Sumner, E.E. 1992. Seesoft – A tool for visualizing line oriented software statistics. *IEEE Transactions on Sotware Engineering.* 18, 11. pp. 957–968.
6. Eick, S. 1994. Graphically displaying text. *Journal of Computational and Graphical Statistics.* 3, 2. pp. 127–142.
7. Good, L., and Bederson, B. 2002. Zoomable user interfaces as a medium for slide show presentations. *Information Visualization.* 1, 1. pp. 35–49.
8. Heckel, P. 1978. A technique for isolating differences between files. *Comm. of the ACM* 21, 4. pp. 264–268.
9. Hirschberg, D.S. 1975. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM.* 18, 6. pp.341–343.
10. Hunt, J.W., and McIlroy, M.D. 1976. An algorithm for differential file comparison. *Bell Laboratories CSTR #41.*
11. Levenshtein, V.I.. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady.*
12. Microsoft. PowerPoint. www.microsoft.com/powerpoint
13. Microsoft PowerPoint Object Model Reference, http://msdn.microsoft.com/office/understanding/powerpoint
14. Moscovich, T., Scholz, K., Hughes, J.F., and Salesin, D. 2004. Customizable presentations. *Technical Report CS-04-16, Computer Science Department, Brown University.*
15. Needleman, S. and Wunsch, C. 1970. *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, J Mol Biol. 48(3):443-53
16. Nelson, L., Ichimura, S., Pedersen, E.R., and Adams, L. 1999. Palette: A paper interface for giving presentations. In *Proceedings of CHI 1999.* pp. 354–361.
17. OpenOffice Impress. www.openoffice.org
18. Parker, A., and Hamblen, J.O. 1989. Computer algorithms for plagiarism detection. *IEEE Transactions on Education.* 32, 2: pp. 94-99.
19. Parker, I. 2001. Absolute PowerPoint: Can a software package edit our thoughts? *The NewYorker.* pp. 76–87.
20. Perlin, K., and Fox, D. 1993. Pad: An alternative approach to the computer interface. In *Proceedings of SIGGRAPH 2003. pp. 57-64.*
21. Salton, G. and McGill, M. J. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc.
22. Viégas, F. B., Wattenberg, M., and Dave, K. 2004. Studying cooperation and conflict between authors with *history flow* visualizations. In *Proceedings of CHI 2004*. pp. 575-582.
23. Zellweger, P.T. 1989. Scripted documents: A hypermedia path mechanism. In *Proceedings of ACM Hypertext 1989.*
24. Zongker, D. E. and Salesin, D. H. 2003. On creating animated presentations. In *Proceedings of the 2003 SIGGRAPH /Eurographics Symp. on Computer Animation*. pp.298-308

Appendix A: Computing String Edit Distance

We use dynamic programming in three different parts of our system (to determine text distances, to find slide correspondences, and to align presentations). Given two strings of lengths *m* and *n* respectively, we construct a cost matrix *D* of size. $(m+1)\times(n+1)$. Entry $(i,j)$of this matrix represents the cost required to convert the first *i* characters of string *1* into the first *j* characters of string *2*. We initialize *D* by filling in the top and left edges of *D* with the numbers *0* to *m* and 0 to *n* respectively. The remaining entries of *D* are computed as follows:

if (*string1*[*i*] = *string2*[*j*]) **then** *COST* = 0; **else** *COST* = 1;

$$D[i,j] = \min\begin{pmatrix} D[i-1,j]+1, // \textit{deletion} \\ D[i,j-1]+1, // \textit{insertion} \\ D[i-1,j-1]+COST, // \textit{substituion} \end{pmatrix}$$

For example, to compare the strings SURVEY and SURGERY, we generate the following matrix:

		S	U	R	G	E	R	Y
	0	1	2	3	4	5	6	7
S	1	**0**	1	2	3	4	5	6
U	2	1	**0**	1	2	3	4	5
R	3	2	1	**0**	1	2	3	4
V	4	3	2	1	**1**	2	3	4
E	5	4	3	2	2	**1**	**2**	3
Y	6	5	4	3	3	2	3	**2**

The bold items represent the minimum cost paths from the start to the end of each string. The algorithm yields a string edit distance of 2 between the strings. Backtracking through the matrix results in the following optimal alignment:

S	U	R	G	E	R	Y
S	U	R	V	E	-	Y

More detailed descriptions of string edit distance can be found in [4,11].