Prescient Paper: Multimedia Document Creation with Document Image Matching

Berna Erol, Jonathan J. Hull, Jamey Graham, and Dar-Shyang Lee
Ricoh Innovations, Inc.
California Research Center
Menlo Park, CA
{ berna_erol, hull, jamey, dsl } @ rii.ricoh.com

Abstract

A system is described for creating paper documents that show images of presentation slides and bar codes that point to a multimedia recording of a presentation that has not yet occurred. An image-matching algorithm applied after *a presentation determines when each slide was displayed. These time-stamps map the bar codes onto commands that control a multimedia player. We describe the system infrastructure that allows us to prepare such prescient documents and the document image-matching algorithm that enables the mapping of bar codes onto the times when slides were displayed. This provides a multimedia annotation tool that requires no electronic device at capture time.*

1. Introduction

Attendees of presentations often try to take notes about key points or descriptions provided by the speaker that are not given on the slides. Often it's difficult to quickly summarize these explanations and the information is lost.

Recognizing this need, presenters make printouts of their slides in advance and give them to the audience so they can take handwritten notes about the talk. This lets people write notes nearby images of the slides and improves their ability to recall what the speaker said.

However, this still does not capture the actual remarks of the presenter and the context in which they were made. This need is addressed by meeting capture systems that save the video, audio, and slide images used when a talk was given [1]. Typically, these systems provide an online interface that lets one search the data they produce [2].

As an alternative, we provide a solution that lets users take handwritten notes on *paper* printouts of slides that are distributed before a presentation. Any time after a talk, they can scan bar codes on the document to invoke an online multimedia player that shows what occurred in the conference room when the slide closest to the bar code was displayed.

The technical challenge is in matching the slide images used to create the paper document with the images of slides saved by a meeting capture system. The association between the printed image and the captured image provides a time stamp in the multimedia recording that is assigned to the bar code nearby the printed image and used to invoke the multimedia player. Of course, this association can only be made after the talk. This delay in the matching process adds to the technical difficulty since printouts of many other presentations, not just the one being processed, might have been made before any given talk occurs.

2. System Infrastructure

The meeting capture system we've developed includes the *Presentation Recorder* shown in Figure 1 [9]. The video output of a laptop is tapped as it's routed to a projector. Key frame images are captured every time the image displayed on the projector changes and those images (i_i) are saved together with the time stamps (t_i) when they were shown. An audio track and a video track are also captured that show the presenter and audience members. The collection of key frame images, time stamps, and audio and video data is called Presentation Recorder document I_i.

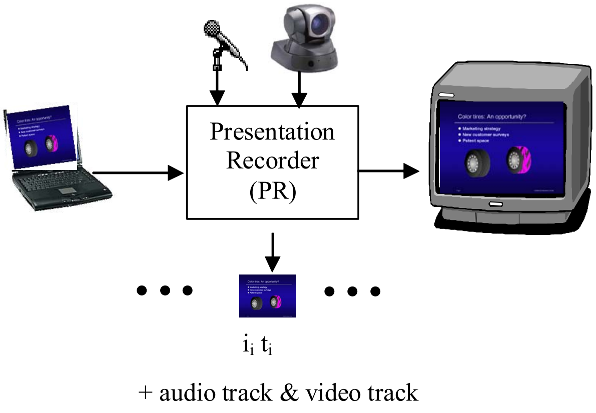

Figure 1. Presentation Recorder – captures images of presentation slides, audio, and video of a presentation.

17th International Conference on Pattern Recognition, Cambridge, U.K., Aug. 2004.

An important design consideration was making this system as easy to use as a projector. This was achieved by tapping the laptop's video output and eliminating the need to install any software on the presenter's computer.

The prescient document preparation system is shown in Figure 2. A presentation document (PowerPoint in our case) is input to a bar code annotation system that creates a paper handout and entries in a database that specify the bar code (bc_k k = 1...m) assigned to the slide ($p_{j,k}$ k = 1...m) in each presentation P_k. A feature extraction algorithm calculates a feature vector f_k for each presentation slide.

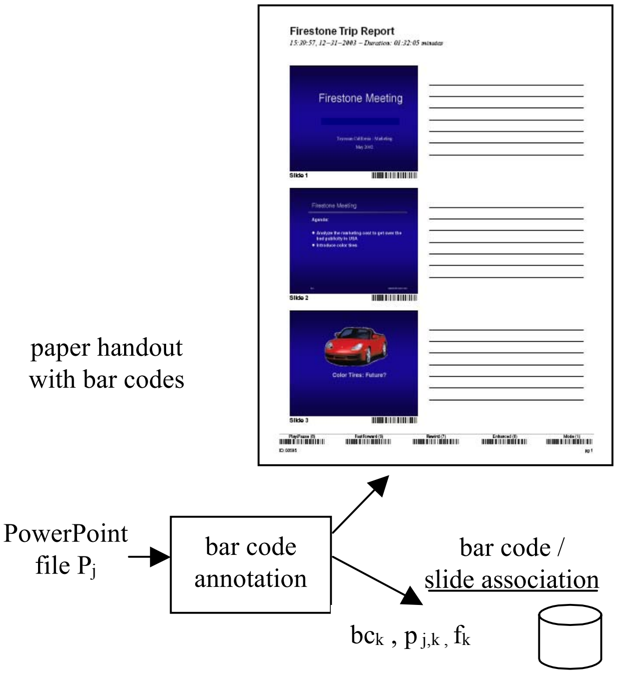

Figure 2. Prescient document preparation system.

3. Document Matching Algorithm

The document-matching algorithm receives a Presentation Recorder document I_i as input and compares it to the database of PowerPoint files. We call this the *presentation-matching* step. It locates every file that could have been used to give the presentation. The next step, called *slide matching,* maps each slide image $p_{j,k}$ in the identified PowerPoint files onto the slide images captured by the Presentation Recorder. This provides the mapping from bar codes in the paper document to time stamps in the audio and video tracks.

The presentation-matching algorithm applies OCR to the Presentation Recorder document and saves the text it outputs together with an indication of the page that the text occurred on. An issue we deal with is the presence of potentially a large number of both duplicate and spurious images in the Presentation Recorder document caused by people going back and forth in their PowerPoint file while they give a talk, the use of custom animations that cause only minor differences to occur in the captured images, and the use of video clips that can cause the Presentation Recorder to capture hundreds of frames.

A pseudo code statement of the presentation-matching algorithm is given below.

```
for every captured slide s in PR document $I_i$
  for every word n-gram w in s
    for every PowerPoint file $P_j$ containing w
      add $P_j$ to the solution set ss
      ++score{$P_j$}
end

for each PowerPoint file $P_j$ in ss
  if (score{$P_j$}/num_words{$P_j$} > $t_1$ &&
    % pages in $P_j$ with > $t_2$ n-grams in $I_i$ > $t_3$)
  then return($P_j$);
end
```

The first step looks up all the PowerPoint files that contain each word n-gram in the Presentation Recorder document I_i and increments a score for that document. The second step considers the PowerPoint files with a minimum percentage of their words in I_i, as specified by t_1, and determines the percentage of pages with more than t_2 of their n-grams in I_i. If this exceeds t_3, then we say this P_j was used to give this presentation.

The presentation-matching algorithm is tolerant of OCR errors in the Presentation Recorder document. Since it contains color jpeg images, we expected the OCR would make many mistakes. However, we observed that because presenters typically choose fonts carefully and use short text phrases, errors were not a significant problem. Also, the requirement that a percentage of the pages in the PowerPoint file be contained in the Presentation Recorder document takes into account duplicate and spurious images. These factors let us set the thresholds liberally.

An additional consideration in the design of the presentation-matching algorithm was the use of existing full text indexes. This was achieved by using word n-grams as query terms to propose potentially matching PowerPoint files.

The slide-matching algorithm determines images in the Presentation Recorder document I_i that match each page in the PowerPoint files located by the presentation-matching algorithm. It uses a combination of string matching on the OCR results for the images that contain text, and edge histogram matching for images that do not contain text.

A pseudo code statement of the slide-matching algorithm is as follows.

```
for each $P_j$ output by presentation-matching
  for each page $p_{j,k}$ in $P_j$
    if ($p_{j,k}$ contains text) then
      time_stamps = string_match($p_{j,k}$, $f_k$, $I_i$);
    else
      time_stamps = image_match($p_{j,k}$, $f_k$, $I_i$);
    update_barcode(time_stamps, $p_{j,k}$);
  end
end
```

The `string_match` function returns the time stamps for all the presentation recorder images within a given string distance of a page in a PowerPoint document. It extracts text from the captured slides by first performing text localization and binarization (described in [7]) and then applies a commercial OCR package. The OCR outputs, for each captured slide image, i_i, a set of text lines, $L_i = \{T_{L1}, T_{L2}, ..., T_{LM}\}$.

The distance between a source slide p_k and a captured slide i_i is found as follows. First, for each text line of p_k, the edit distances to the each line of i_i are computed. Since i_i may contain many extra characters caused by OCR errors, here we ignore the deletions when computing the edit distance. Next, each of these distances are scaled by the length of the ground truth string. For each processed line, the minimum of these distances are summed to obtain the distance from p_k to i_i, i.e.,

$$d(p_k, i_i) = \sum_{n=1}^{N} \min_{T_{Lm} \in L_i} \left\{ \frac{\ell_{TPn} - D_e(T_{Pn}, T_{Lm})}{\ell_{TPn}} \right\},$$

where T_{Pn} is the n^th text line in p_k, T_{Lm} is the m^th text line in i_i, N is the number of text lines in p_k, D_e is the edit distance, and ℓ_{TPn} is the length of T_{Pn}. The $d(p_k, i_i)$ measure is asymmetric, i.e. the distance from p_k to i_i is not equal to the distance from i_i to p_k. We define the final distance between two slides as the maximum of these two distances:

$$D(p_k, i_i) = \max\{ d(p_k, i_i),\ d(i_i, p_k) \}.$$

The `image_match` function returns the time stamps for all the presentation recorder images within a given L2 distance of a page in a PowerPoint document. It extracts an edge histogram as follows. First, a modified Sobel operator is applied to the image to obtain edge magnitudes. An edge is detected only if the edge magnitude is larger than a threshold and an edge direction has changed.

After horizontal and vertical edge pixels are extracted, the image is divided into N× M segments and the number of horizontal and vertical edge pixels in each segment are stored in the horizontal, H_{hor}, and vertical, H_{ver}, edge histograms as follows:

$$H_{ver}(n,m) = \frac{1}{S_M S_N} \sum_{y=mS_M}^{(m+1)S_M} \sum_{x=nS_N}^{(n+1)S_N} \text{verticalEdge[x][y]},$$

where N is the number of horizontal segments and M is the number of vertical segments, *verticalEdge[x][y]* is the detected edge value at location (x,y), S_N and S_M are the height and width of segment (*n,m*) in pixels and are computed by S_N=ImageWidth/N and S_M=ImageHeight/M. Horizontal edge histogram, $H_{hor,}$ is found in a similar way.

The resulting feature vector includes N× M vertical edge histogram bins and N× M horizontal edge histogram bins. Two global edge features are also included in the feature vector, which are computed by summing the values of the vertical and horizontal edge histograms separately and normalizing them by N× M as follows.

$$G_{ver} = \frac{1}{MN} \sum_{m=0}^{M} \sum_{n=0}^{N} H_{ver}(n,m).$$

The resulting feature vector is:

$$\vec{F} = [G_{hor}, G_{ver}, H_{hor}(0,0), ..., H_{hor}(n,m), H_{ver}(0,0), ..., H_{ver}(n,m)].$$

The distance between the edge histogram feature vectors, $\vec{F}1$ and $\vec{F}2$, are found by computing the L2 norm of their difference.

4. Experimental Results

In this section we present experimental results for both the presentation-matching and slide-matching algorithms. For presentation matching, we compared the performance of n-gram word matching (with n=1 and n=2) to matching word sets with a tf-idf (term frequency-inverse document frequency) measure. The experiments were performed using 5 presentations captured with the *Presentation Recorder* and a database of 51 PowerPoint presentation files. Table 1 shows the results. These were obtained with the threshold, t_3=0.3. The confidence value, C_m, is defined as, $C_m = \frac{(S_1 - S_2)}{S_1}$, where S_1 and S_2 are the matching scores obtained for the best matched and second best matched slides, respectively. Recall was 1 for all cases. Overall, n-grams with n=2 provided the best performance. Precision was 1 in every case and confidence values were consistently higher than those produced by the TFIDF measure.

Table 1. Performance of presentation matching algorithm.

Pres Rec. doc.	*N=1*		*N=2*		*TFIDF*	
	prec	C_m	*prec.*	C_m	*prec.*	C_m
1	0.2	0.42	1	0.93	1	0.72
2	0.14	0.37	1	0.87	1	0.70
3	0.25	0.18	1	0.83	1	0.33
4	0.33	0.39	1	0.82	1	0.87
5	0.25	0.27	1	0.86	1	0.78

After the correct presentation file is identified, the slide-matching algorithm uses string and edge matching techniques to determine when each slide was shown. Here, the text and edge histograms are extracted from each PowerPoint slide and they are matched with the slides captured by the *Presentation Recorder.* Precision and recall rates obtained by querying 348 slides in 7 presentations are presented in Table 2. As can be seen from the table, recall rate is quite high. In our application, this means that when a user scans a barcode on his/her handouts, 98% of the time he/she will access the multimedia recording of the speaker talking about that particular slide.

Table 2. Performance of slide matching algorithm.

Presentation	Number of slides	*Precision*	*Recall*
1	69	1	0.96
2	38	1	0.97
3	49	1	1.00
4	72	1	1.00
5	120	1	0.98
ALL	**348**	**1**	**0.98**

5. Related Work

Conference rooms and lecture halls with automated capture systems provide fertile ground for exploring new methods for multimedia note taking. Prescient Paper is unique in several ways compared to the prior art. In one class of method, users compose handwritten notes on PDA's that are uploaded to a server and cross-indexed with the recorded lecture [5][11]. Another system provides a special-purpose client application on a tablet PC that lets a user annotate a meeting and create a web page that includes links to multimedia objects accessible by post-hoc network access [4].

These systems contrast sharply with our emphasis on letting a user take notes with a normal pen or pencil on a paper document without any electronic device. This makes multimedia note taking almost the same as traditional paper-based note taking. This eliminates any reluctance one might have in using an electronic device during a presentation and provides a retrieval interface that can be used either with or without a PC. That is, just the handwritten notes can be used when appropriate. But when additional information is needed, the user only need scan the nearest bar code to see what was said when the slide was displayed. This will be especially valuable in pervasive environments where video content will be delivered to wireless devices [6], like the combination bar code reader/iPaq used with our system [9].

The document image-matching algorithm we use combines a text-based approach with an image-based technique. The text-based method uses the recognized strengths of string matching [10] to provide a highly accurate match from imperfect page images to images extracted from PowerPoint files. The image-based algorithm was incorporated because of the unique nature of PowerPoint documents that might contain pages with no text on them at all, just image data. Also, spurious non-document images can be interspersed within a Presentation Recorder document. This requirement is similar to that faced by layout comparison algorithms such as [8].

6. Discussion and Conclusions

A novel document image-matching algorithm was described that enables the production of paper documents that refer to multimedia data before it exists. The document matching technique overcomes the need to install any software on a presenter's laptop, which was a significant barrier to adoption. Over three years experience in recording over 200 presentations with the system described in this paper reinforced the need for an annotation system, like the one described here, which can be used without any electronic device. The infrastructure described in this paper is fully implemented in our lab and we are currently exploring how it can help people use multimedia data as part of their regular work practice.

7. References

[1] G.D. Abowd, et al., "Teaching and Learning as Multimedia Authoring: The Classroom 2000 Project," Proc. of ACM Multimedia, pp.187-198, Nov. 1996.

[2] J.A. Brotherton, J.R. Bhalodia, and G.D. Abowd, "Automated Capture, Integration, and Visualization of Multiple Media Streams," Proc. IEEE Multimedia'98.

[3] M. Cheriet, S. Naoi, and C. Y. Suen, "Automatic Filter Selection Using Image Quality Assessment," Int. Conf. on Document Analysis and Recognition, 508-512, Aug. 2003.

[4] P. Chiu, A. Kapuskar, S. Reitmeier and L. Wilcox, "NoteLook: Taking Notes in Meetings with Digital Video and Ink," Proc. of ACM Multimedia'99, Nov. 1999.

[5] R.C. Davis, J.A. Landay, et.al, "NotePals: Lightweight note sharing by the group, for the group," Proc. CHI'99, pp.338-345, 1999.

[6] D. Doermann, N. Parekh, and V. Rautio, "Video Analysis Applications for Pervasive Environments," 1st Int. Conf. Mobile and Ubiquitous Multimedia, Dec. 2002, 48-55.

[7] B. Erol, J. J. Hull, and D. S. Lee, "Linking presentations with their symbolic source documents: algorithm and applications," ACM Multimedia 2003, Berkeley, CA.

[8] J. Hu, R. Kashi, and G. Wilfong, "Document Image Layout Comparison and Classification," Int. Conf. on Document Analysis and Recognition, 285-288, India, Sept. 1999.

[9] D. S. Lee, B. Erol, J. Graham, J. J. Hull and N. Murata, "Portable meeting recorder," ACM Multimedia 2002, France, Dec. 1-6.

[10] D. Lopresti, "String Techniques for Detecting Duplicates in Document Databases," Int. Journal on Document Analysis and Recognition, v. 2 no. 4, June 2000, 186-199.

[11] K.N. Truong and G.D. Abowd, "StuPad: Integrating Student Notes w/ Class Lectures," Proc. CHI'99, 208-209.