PaperDiff: A Script Independent Automatic Method for Finding The Text Differences Between Two Document Images

Sitaram Ramachandrula, Joshi Gopal Datt, Noushath. S, Pulkit Parikh, Vishal Gupta

HP Laboratories
HPL-2008-130

Keyword(s):

paperDiff, document segmentation, skew detection, text line extraction, dynamic programming, image matching, correction

Abstract:

In this paper, we introduce a novel concept called PaperDiff and propose an algorithm to implement it. The aim of PaperDiff is to compare two printed (paper) documents using their images and determine the differences in terms of text inserted, deleted and substituted between them. This lets an end-user compare two documents which are already printed or even if one of which is printed (the other could be in electronic form such as MS-word *.doc file). The algorithm we have proposed for realizing PaperDiff is based on word image comparison and is even suitable for symbol strings and for any script/language (including multiple scripts) in the documents, where even mature optical character recognition (OCR) technology has had very little success. PaperDiff enables end-users like lawyers, novelists, etc, in comparing new document versions with older versions of them. Our proposed method is suitable even when the formatting of content is different between the two input documents, where the structures of the document images are different (for e.g., differing page widths, page structure etc). An experiment of PaperDiff on single column text documents yielded 99.2% accuracy while detecting 135 induced differences in 10 pairs of documents.

External Posting Date: October 6, 2008 [Fulltext] Approved for External Publication
Internal Posting Date: October 6, 2008 [Fulltext]
Submitted to the Eighth IAPR International Workshop on Document Analysis and Systems (DAS 208) September 17-19, 2008

PaperDiff: A Script Independent Automatic Method for Finding The Text Differences Between Two Document Images

Sitaram Ramachandrula, Gopal Datt Joshi[*], Noushath.S, Pulkit Parikh and Vishal Gupta[†]
Hewlett-Packard Labs India
Bangalore, India
{sitaram, noushath.s, pulkit.parikh}@hp.com

Abstract

*In this paper, we introduce a novel concept called PaperDiff and propose an algorithm to implement it. The aim of PaperDiff is to compare two printed (paper) documents using their images and determine the differences in terms of text inserted, deleted and substituted between them. This lets an end-user compare two documents which are already printed or even if one of which is printed (the other could be in electronic form such as MS-word *.doc file). The algorithm we have proposed for realizing PaperDiff is based on word image comparison and is even suitable for symbol strings and for any script/language (including multiple scripts) in the documents, where even mature optical character recognition (OCR) technology has had very little success. PaperDiff enables end-users like lawyers, novelists, etc, in comparing new document versions with older versions of them. Our proposed method is suitable even when the formatting of content is different between the two input documents, where the structures of the document images are different (for e.g., differing page widths, page structure etc). An experiment of PaperDiff on single column text documents yielded 99.2% accuracy while detecting 135 induced differences in 10 pairs of documents.*

1 Introduction

There are several occasions when we compare: (a) two printed versions of a document or (b) a printed document and an electronic version of it to identify any differences/modifications. A good example of this is while comparing two versions of a legal agreement. In the legal agreement process, the text content of the agreement is normally exchanged first (e.g., by email/print, etc) between all the parties involved, and finally the agreed content is printed (sometimes on a legal stamp paper) and all the parties sign it. In this scenario the parties who are signing the agreement printed by others have to quickly check/verify whether the text in the printout is exactly the same as in the agreed version or were there any subsequent changes carried out. Some of the legal agreements run into many pages, making this verification process tedious and error-prone if done manually.

An obvious automatic method of solving this problem is using optical character recognition (OCR) technology, where the images of documents are converted into text (e.g., ASCII) and using *string matching* algorithms exact textual differences between them could be determined. The limitations of this approach are: a) for many of the scripts in the world OCR engines are not available, b) OCR engines even where available are not 100% accurate, c) for documents with multiple language scripts or symbols, the text conversion module using OCR technology is even more complex and is still a research problem.

In this paper, we propose an automatic method of finding differences between two document images without using an OCR. In our method the fundamental processing block is comparing/matching two small images. The small images we are referring to are typically the individual word images extracted from the input document. In this work we refer these images as DBUs. By intelligently applying this image matching iteratively using well known dynamic programming of optimization technique, it is possible to find all the textual differences between the document images. As our method is independent of OCR it is suitable for any scripts, symbol strings, etc.

We are not aware of any earlier work of document comparison with an intention to determine the exact text differences. However, there has been considerable work in duplicate document detection [8] and document image retrieval areas [9] which use similar algorithms. There the intention is to absorb small differences in text and yet retrieve documents or detect the duplicates. In our problem, the main

[*]Email: gopal@research.iiit.ac.in
[†]Email: vishal.gupta@gmail.com
(Formerly with HP Labs India)

intention is to finely detect all these text changes. In Spitz approach [8] where character shape codes (CSC) are used to represent text, two different words can have same CSC and may go undetected. In Hull's method of using number of character in each word [9] as an intermediate representation, two different or consecutive words may be of same character lengths thus may get misaligned. These algorithms cannot be directly applied for document comparison application as these algorithms have been designed to absorb these variation/changes and yet achieve their respective goals. One important difference in our method is we use a distance metric between two DBU images as a fundamental processing block.

Rest of the paper is organized as follows: The problem that we are addressing and the proposed solution are explained in section-2 and section-3 respectively. Section-4 gives some experimental results and concluding remarks are drawn in section-5.

2 Problem Statement

In general the problem being addressed can be stated as follows: Given a printed document, Document-1 in any script and another printed or electronic version of it, Document-2. The need is to find the textual differences between Document-1 and Document-2 and render those differences to the user either through an appropriate user interface or on paper. The method should also work when there are formatting differences between the two documents (e.g., differing page width, differing layout, etc).

3 Proposed solution

The proposed solution relies on the properties common to many scripts and does not use properties specific to any particular script. Many of the scripts are written in (i) straight lines (either vertically/horizontally), (ii) line after line, (iii) paragraph after paragraph and (iv) in specific order (i.e., left-to-right or right-to-left or top-to-bottom). The proposed methodology is explained here for scripts written from left to right. For scripts written in a different order, the same method could be applied with suitable alterations. For mixed script documents, with different write orders, for eg., English (left to right) mixed with Japanese (top to bottom), the proposed algorithm does not work automatically.

Our solution to tackle document comparison is depicted in Figure 1. It relies on existing techniques of document layout analysis [1, 2, 7] and read order determination in the Stage-I (see figure 1) for extracting sequences of document basic units (DBUs). The DBUs being referred here are images of one word (refer Figure 2). The stage-I transforms the two input document images into two ordered

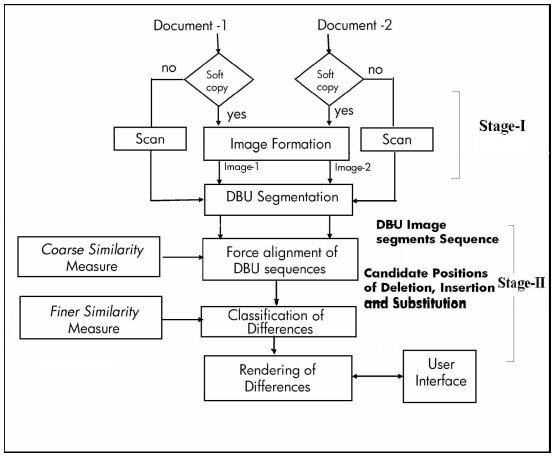

Figure 1. Complete work-flow of PaperDiff

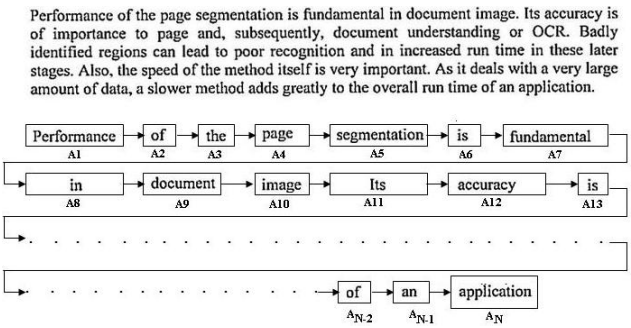

Figure 2. An example paragraph image (top) and its DBU sequences (bottom)

sequences of DBU image segments $A = A_1, A_2, \cdots, A_m$ and $B = B_1, B_2, \cdots, B_n$, where A_i and B_j are individual DBU images as illustrated in Figure 2. More details of this stage are given in section 3.1 and 3.2.

Once the sequences of DBUs are extracted from both the input documents, the Stage-II compares these two sequences (strings) using well known dynamic programming technique. This step determines the DBUs that are inserted, deleted or substituted between the two documents after force aligning them. The details of this are given in section 3.3. This stage also needs previously mentioned fundamental processing block of matching two DBU images. This matching computes the distance scores (similarity) between all pairs of DBU images, each taken from two input sequences. This process is explained in detail in section 3.4.

In subsequent subsections, aforementioned steps are elaborated.

3.1 Removal of Figures

In our current work, we have assumed that the two input document images are free of graphics etc., and only textual content is present. Since the PaperDiff is mainly intended for the applications which predominantly includes text content, this is a safe assumption to start testing the concept with only textual content. However, in real world applications it is not always possible to enforce text only documents. In the legal agreement example we gave, many times there is a presence of figures, typically of legal stamp apart from the text content in the documents. To automatically remove figures or to retain only the textual regions in the given document there exist several methods in literature [3, 5]. One could potentially apply these methods to extract the textual regions in the document.

3.2 DBU Sequence Extraction

A textual document image can be divided into an ordered sequence of image segments of DBUs (as in Figure 2) using page segmentation techniques such as X-Y cut [7], Docstrum [1] etc. There are many different approaches in literature for this stage [2, 3, 4]. The next stage of PaperDiff needs the DBU sequences arranged in right order (which depends on the class of scripts). At this stage we assume prior knowledge of class of scripts on document, i.e. whether the read order is horizontal (left-to-right/right-to-left) or vertical (top-to-bottom). It may be possible to automatically detect the class of scripts on the document, i.e. horizontal or vertical. In our implementations, we have also assumed that the document is of single column format and reading order is from left to right. However, the proposed solution can be easily altered for any reading order.

3.3 DBU (Word) Alignment Using Dynamic Programming

This sub-section explains the main theory in achieving the goal of this paper. Let $A = \{A_1, A_2, \cdots, A_N\}$ and $B = \{B_1, B_2, \cdots, B_N\}$ are the DBU sequences extracted from two inputs documents. The main problem is to find differences between the two sequences **A** and **B**. We propose to achieve this by force aligning two DBU sequences which ideally should pair similar or identical DBU images occurring in both the input sequences. Then detecting any differences like DBU insertions and deletions when there is a misalignment i.e., a blank paired to a DBU. The differences between sequences of DBU (word) images **A** and **B** may be computed optimally (in time proportional to $m \times n$, where m is the length of the first sequence and n is the length of the second sequence) by using Dynamic Programming (DP) for this force alignment. DP here needs a definition of local distance, i.e., distance between any pair of DBUs and also cost for pairing a DBU with a blank, which is equivalent to deletion or insertion of DBUs.

The problem here is to find the optimal alignment path (best path in a trellis) which has a minimal sum of distances between all the paired DBUs including the costs of possible skipping the pairing i.e., insertion or deletion costs along the given path. This can be implemented iteratively by extending the best partial path at each iterative step given in the algorithm (Box-1) which in the end minimizes the total distance between the two sequences and finds the best alignment.

Let D_{ij} be the cost/distance of the best aligned partial path between sequences A and B (see figure 3), until DBUs A_i and B_j from the beginning; and let $w(A_i, B_j)$ be the local distance (weight) between two word image units A_i and B_j. For similar DBU images, weight is smaller and larger for dissimilar DBUs (refer figure 4). The complete steps involved in this dynamic programming algorithm are explained in Box-1.

For each execution step in Eq.2, depending on which of the three quantities is minimum, we assign a *status* to the current (i,j) pair - *Insertion*, *Deletion* or *No-change*. These status labels, stored in a 2-D table $S_{i,j}$, allow us to backtrack the best path. This is done using path variable $P_{i,j}$, as detailed in the third step of Box-1.

Here $w(A_i, \Phi)$ is the cost associated with the insertion of A_i in **A** (or deletion of unit A_i in **B**) and $w(\Phi, B_j)$ is the cost of the insertion of B_j in sequence **B** (these are fixed values where Φ indicates absence of DBU). And $w(A_i, B_j)$ is the local distance between image segments A_i and B_j (this also works as the cost for substitution of A_i with B_j if both are different words). Finally, the total dissimilarity score between two sequences becomes the minimal possible sum of local distances between all DBUs paired including possible insertions and deletions from beginning to end. This represents an optimal alignment of sequences **A** and **B** (Figure 3). Consequently, the locations of differences after alignment can be identified in the backtracking step whenever a DBU got paired with a blank i.e., Φ.

In the aligned sequences, a few partially similar DBUs get paired due to a smaller local distance. In order to identify these smaller differences between word images, a fine distance analysis based on a length normalized threshold is performed on all paired DBUs after alignment (e.g., *complete* changed to *incomplete* or *horse* to *horses*, etc is treated as a substitution). Thus, the differences in the two documents are reported in the form of insertions, deletions and substitutions

3.4 Local Distance: Image Matching

The essence of local distance $w(x, y)$ we have used between two DBU images in our implementation of the pro-

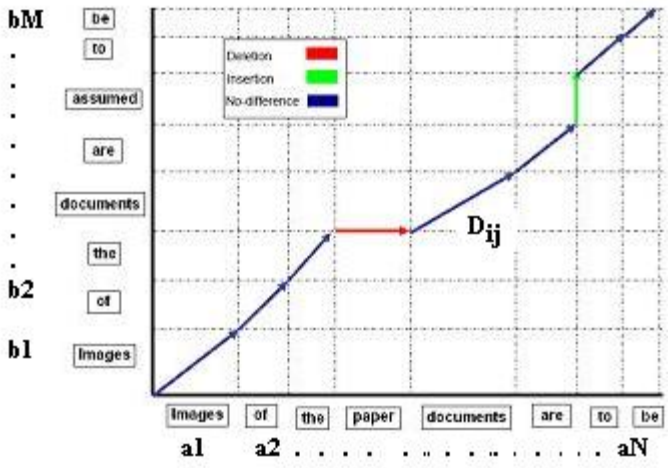

Figure 3. An example of optimal alignment of word sequences

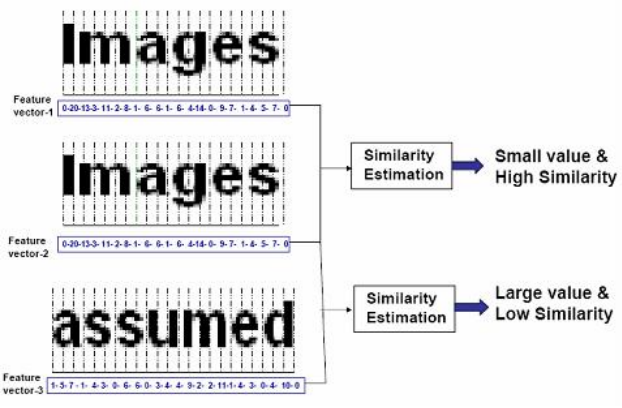

Figure 4. Illustration of local distance metric used between two word images

posed dynamic programming algorithm, is illustrated in figure 4. Let I_A and I_B be two DBU segments of length m_1 and m_2. Then the image distance we have used is defined as

$$w(I_A, I_B) = \frac{\sum_i (f_i(I_A) - f_i(I_B)}{max(m_1, m_2)} \quad (1)$$

$\forall i = 1, 2, \cdots, max(m_1, m_2)$.
Here, f_i is the number of black pixels in the i^{th} column (columns are depicted in figure 4) of a DBU (word) image. This is the coarse similarity measure we mentioned in Fig.1. This does not give accurate local distance as many times DBUs of different lengths are matched. But this is computationally simple and good enough for force alignment. We have used various distance measures at a higher resolution to identify finer differences in the coarsely aligned pairs of DBU (word) images, e.g. 2D correlation of DBU images, dynamic time warping of each DBU pairs using their $f_i(I)$ sequences. As the DBUs are matched by treating them as images (irrespective of their script information), this method works for any script, hence we call our method as script independent.

4 Experimental Evaluation

In this section, we report the preliminary experimentation results that we obtained using simple X-Y cut page segmentation algorithm [7].
In our experiment, first the document is de-skewed using well known skew correction technique [6] and then individual text line images are segmented using horizontal projection profile of the image. Next, vertical projection profile of each of the segmented line image is computed and the DBU image segments are extracted. The alignment of two sequences of DBU segments is achieved using the proposed algorithm along with the local distance discussed earlier.

In a database of 10 single column documents each of one page length, with 12 pt text font size, having 135 induced differences the algorithm detected 99.2 % of differences (the only 1 error we got is due to DBU segmentation error). However, these are very few false alarms due to page segmentation issues. The documents we tested include text contents in different scripts like English, Kannada, Tamil, Telugu, Hindi etc. These languages form an important subset of Asian languages. We also included multi-script documents containing text from all these languages. Though the alterations needed to our technique for different classes of scripts is minimal, we would like to emphasize the fact that the proposed solution to realize *PaperDiff* is basically script independent. As our method is independent of OCR it is suitable for multi script documents and thus it behaves like a script independent approach as long as the reading orders of different scripts in a the document are same. Of course, the approach may not work when the document contains scripts of mixed reading orders.

While comparing an electronic document with a printed document, the electronic document is first converted to an image file (i.e. synthetic image) using MS Word printer options and then the same method is used. The figure 5 shows an example of two simple documents with induced differences (deletion/insertion of some words) and their results. The differences found between two input documents are shown with black patches at corresponding positions.

We plan to use more advanced page segmentation techniques like Docstrum [1], area Voronoi method [2] etc to work on complex documents such as the one with multi columns, graphics, varying font size etc.

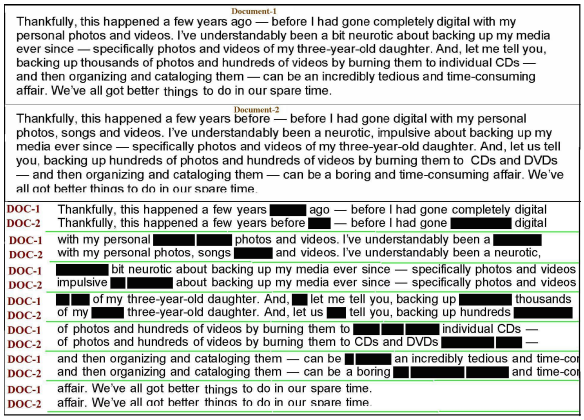

Figure 5. An example of two documents (top and middle) with induced differences and their results (bottom). Black blocks indicate the points of difference

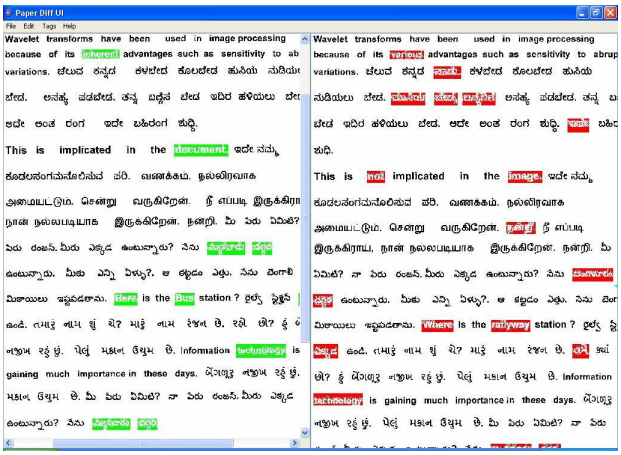

Figure 6. A PC screen shot of PaperDiff UI. The colored words indicate the points of difference

5 Conclusions and Discussions

We have proposed a novel concept/problem of comparing two documents to find text differences in them. Also our proposed algorithm for this is achieving encouraging results. It is possible to extend this to more complex documents by applying suitable page segmentation techniques.

It is not possible to use well known image registration technology for this problem as that is only suitable when the structure of input images do not differ much. In our solution, the structure can drastically change by even inserting a paragraph, changing page width, etc, and this affects all subsequent pages because of text re-flow.

We have also got a user-friendly interface that provides a rendering of both images side-by-side, allowing a user to quickly navigate between the two documents. A PC screen shot of the final alignment result of our method on two mixed script documents is shown in figure 6, where the highlighted words indicate insertions or deletions between the two documents. In future, we are aiming to handle documents with more complex layout - graphics, handwritten annotations, single/double columns etc, using suitable page segmentation techniques. Also we are working at very fast alignment algorithms for large documents.

References

[1] Lawrence O' Gorman, *The Document Spectrum for Page Layout Analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.15, No.11, November 1993, pp.1162-1173.

[2] Koichi Kise, Aknori Sato and Motoi Iwata, *Segmentation of Page Images Using the Area Voronoi Diagram*, Computer Vision and Image Understanding, Vol.70, No.3, June 1998, pp. 370-382.

[3] S. Basu, C. Chaudhuri, M. Kundu, M. Nasipuri and D. K. Basu, *Text Line Extraction From Multi-Skewed Handwritten Documents*, Pattern Recognition, Vol.40, 2007, pp.1825-1839.

[4] Debashish Niyogi, *A Knowledge Based Approach to Deriving Logical Structure From Document Images*, Ph.D thesis, State University of New York at Buffalo, 1994.

[5] Farshad Nourbakhsh, Peeta Basa Pati and A. G. Ramakrishnan, *Text Localization and Extraction from Complex Gray Images*, ICVGIP 2006, pp.776-785.

[6] C. -H. Chou, S. -Y. Chu, and F. Chang, *Estimation of Skew Angles for Scanned Documents Based on Piecewise Covering By Parallelograms*,Pattern Recognition, Vol.40, 2007, pp.443-455.

[7] G. Nagy, S. Seth and M. Vishwanathan, *A Prototype Document Image-Analysis System for Technical Journals*, Computer, Vol.25, 1992, pp. 10-22.

[8] A. Lawrence Spitz, *Duplicate Document Detection*, Document Recognition IV. Proc. SPIE 3027, 1997, pp. 8894

[9] J. J. Hull, *Document Image Matching and Retrieval with Multiple Distortion-Invariant Descriptors*, in Document Analysis Systems, A.L.Spits and A.Dengel, eds., World Scientific, Singapore, 1995, pp.379-396.

Box−1

Algorithm:*DBU sequence alignment using dynamic programming*
Input: Two sequences of DBU (word) images of two documents.
Output: Aligned set of DBUs (words).
Steps:

1. **Initialization Step**:

$$\begin{aligned} D_{i,0} &= D_{i-1,0} + w(A_i, \Phi)\ 1 \leq i \leq m \\ D_{0,j} &= D_{0,j-1} + w(\Phi, B_j)\ 1 \leq j \leq n \\ D_{0,0} &= 0 \end{aligned}$$

   Where D_{ij} is the cost/distance of the best aligned partial path between sequences A and B.

2. **Recursion Step**:

$$D_{ij} = min \begin{cases} D_{i-1,j} + w(A_i, \Phi) \\ D_{i-1,j-1} + w(A_i, B_j) \\ D_{i,j-1} + w(\Phi, B_j) \end{cases} \tag{2}$$

$$S_{i,j} = \begin{cases} I, if\ D_{i,j} = D_{i-1,j} + w(A_i, \Phi) \\ N, if\ D_{i,j} = D_{i-1,j-1} + w(A_i, B_j) \\ D, if\ D_{i,j} = D_{i,j-1} + w(\Phi, B_j) \end{cases} \tag{3}$$

   Where $1 \leq i \leq m\ 1 \leq j \leq n$

3. **Backtracking Step**:
   Let $P_{i,j}$ be the path of DBU pairs ending at (i, j).
   Start: $i = m$, $j = n$, $P_{0,0} = \Phi$

$$P_{i,j} = \begin{cases} P_{i-1,j-1} + [A_i, B_j], if\ S_{i,j} = N \\ P_{i-1,j} + [A_i, \Phi], if\ S_{i,j} = I \\ P_{i,j-1} + [\Phi, B_j], if\ S_{i,j} = D \end{cases} \tag{4}$$