Aligning transcript of historical documents using dynamic programming

Irina Rabaev∗, Rafi Cohen∗, Jihad El-Sana and Klara Kedem

Department of Computer Science,
Ben-Gurion University, Beer-Sheva, Israel

ABSTRACT

We present a simple and accurate approach for aligning historical documents with their corresponding transcription. First, a representative of each letter in the historical document is cropped. Then, the transcription is transformed to synthetic word images by representing the letters in the transcription by the cropped letters. These synthetic word images are aligned to groups of connected components in the original text, along each line, using dynamic programming. For measuring image similarities we experimented with a variety of feature extraction and matching methods. The presented alignment algorithm was tested on two historical datasets and provided excellent results.

Keywords: alignment, dynamic programming, historical documents, profile-based features, GSC features, HOG features, LBP features

1. INTRODUCTION

An ongoing considerable effort for digitizing historical manuscripts have produced huge datasets. In such datasets the documents are saved as images, which makes the text processing slow and complicated. Sometimes an ASCII transcription is supplied together with the document’s image. A mapping (alignment) between the words in the transcription and the words in the image can simplify and accelerate accessing and processing the manuscripts. In addition to browsing and searching the document images, alignment provides an automatic way for ground truth generation, which in turn can be used to evaluate various document retrieval and recognition algorithms.

The presented alignment method is based on dynamic programming. First, synthetic line images are generated using the given transcription and the cropped representatives of each letter. Then each line in the original document is aligned with its synthetic counterpart, where synthetic words are matched against groups of connected components from the original line. To measure the similarities of two images (the synthetic word and the candidate image patch from the original line) various distance functions and feature extraction schemes can be applied. In this work we experimented with well-known classes of features, the profile-based, GSC, HOG and LBP features, together with appropriate distance functions.

The presented algorithm was tested on two datasets of historical documents and demonstrated excellent results.

∗ These authors have contributed equally to this work

Further author information:
E-mail: rabaev, rafico, el-sana, klara@cs.bgu.ac.il
Corresponding authors: I.Rabaev, R.Cohen
2. RELATED WORK

In this section we briefly overview the related work for aligning text in a document’s image with its transcription. Tomai et al.\textsuperscript{1} combine a word recognition with word mapping. In their approach, each line is assigned a lexicon (subset of words) based on the line transcription. Several hypotheses for line segmentation are generated and word recognition is used to assign to each word image from the hypothesis a transcript word with a certain confidence. Dynamic programming is then used to get the highest scoring segmentation. The alignment is updated each time a sequence of consecutive words with high confidence (an anchor) is found. Huang and Srihari\textsuperscript{2} also combine word recognition and dynamic programming to perform the alignment task. The alignment is performed at page level. First, a set of global anchor words is found. Then, the document is divided into subsequences, where each subsequence is defined by the set of all the words between two consecutive anchors. Each subsequence in turn is treated as a shorter document. Kornfield et al.\textsuperscript{3} map segmented word images to the transcription using Dynamic Time Wrapping (DTW). In addition, the authors have experimented with several cost functions to compute the distance between a segmented word image and a transcription word in the DTW algorithm. Jose et al.\textsuperscript{4} present the True Dynamic Time Warping (TDTW) algorithm, which cares the segmentation errors by concatenating transcript words and comparing with a single word image, and concatenating word images to compare them with a single transcript word. A sum of the dissimilarity score from a word-model recognizer and the Levenshtein distance between the recognized word and the transcript word is used as a cost metric in the TDTW. Rothfeder et al.\textsuperscript{5} use word level Hidden Markov Models (HMM), where word images are treated as hidden variables and the observations are the feature vectors extracted from the images. Fitcher et al.\textsuperscript{6} presented an alignment system based on character Hidden Markov Models. A page HMM is created according to the given transcription and a number of spelling variations, and is matched against a sequence of features extracted from the text line images of the page. Hassner et al.\textsuperscript{7} synthesize a reference image of the transcript line using a suitable reference font. The synthetic line is matched against the cropped line using a modified SIFT flow. This matching is performed at pixel level by transforming the pixels of the letters in the synthetic image to those in the historical image. Azawi et al.\textsuperscript{8} use an adapted weighted finite state transducers for generating correspondences between the document image and the transcription, where OCR errors and different transcription layouts may occur.

3. OUR METHOD

In a pre-processing step, the document images are segmented into lines by a method recently developed in our group by Cohen et al.\textsuperscript{9} Once the lines are found, they are segmented and binarized (see Figure 1). This reduces the objective to aligning an original line with its synthetic counterpart. Generating synthetic handwriting for training classifiers has been studied extensively in the literature.\textsuperscript{10–12} In this paper, we use a straightforward method for generating synthetic lines. Given the transcript of a line, which is composed of a sequential set of words, we generate the synthetic lines as follows. We crop for each letter in the alphabet a representative character from the historical document image. Due to the fact that the handwriting is usually cursive, we seek characters that are not connected to their neighbours and crop them to avoid false ligatures in the generated synthetic line. Letters that do not have ascenders and descenders are placed within two imaginary lines, baseline and midline. Ascenders are placed above the baseline, and descenders are placed such that their top is below the midline. We space the words using the average distance between words in the original document. See Figure 2 for an illustration.

Due to the manner of line synthesis, we can expect that:

1. Both the line image and the synthetic line contain exactly the same word sequence;
2. There can be only small difference between the size of the original word and the corresponding synthetic word.
3.1 Description of the algorithm

The alignment is done using dynamic programming which matches the words in the synthetic line to groups of connected components in the original line.

Let $cc_1, cc_2, \ldots, cc_m$ be the connected components in the original line, $w_1, w_2, \ldots, w_n$ be the words in the synthetic line, $I_{cc_1, \ldots, cc_i}$ be the image defined by the connected components $cc_1, cc_2, \ldots, cc_i$, and $I_{w_j}$ be the image of the word $w_j$. We generate a table $T$ of size $m \times n$, where $T(i, j)$ is defined as the cost of aligning connected components $cc_1, cc_2, \ldots, cc_i$ with the words $w_1, w_2, \ldots, w_j$. $T(i, j)$ is computed using the recurrence formula detailed in Eq. (1), where $\text{dist}(\cdot, \cdot)$ measures the distance between two images.

$$T(i, 1) = \text{dist}(I_{cc_1, \ldots, cc_i}, I_{w_1}),$$
$$T(i, j) = \min_k \left(T(k, j - 1) + \text{dist}(I_{cc_{k+1}, \ldots, cc_i}, I_{w_j})\right), \quad k < i,$$

Thus, the cost of aligning the connected components $cc_1, \ldots, cc_i$ with the words $w_1, \ldots, w_j$ is equal to the minimum cost, over all $k < i$, of aligning $cc_1, \ldots, cc_k$ with $w_1, \ldots, w_{j-1}$ plus the distance between the images $I_{cc_{k+1}, \ldots, cc_i}$ and $I_{w_j}$. When computing the formula in Eq. (1), we can discard the cases for which the size of the image $I_{cc_{k+1}, \ldots, cc_i}$ differs significantly from the size of the image $I_{w_j}$. In our implementation, we process $I_{cc_{k+1}, \ldots, cc_i}$ only if its width is in the range of 0.5 to 1.5 times the width of $I_{w_j}$, which is reasonable for handwritten text. At the end of the algorithm, an optimal alignment can be retrieved by backtracking the table from $T(m, n)$ along the minimal path.
The distance/similarity of two images can be measured in various ways, and often involves the extraction of representative features. In this work we experimented with a number of well-known feature extraction and matching methods, a brief description of which is given in Secs. 3.2 – 3.5.

### 3.2 Profile-based features

We concentrated on the following profile-based features, most of which have been recommended in the literature.\(^{13–15}\) For each pixel column in the image \(I\) we calculate the following features:

1. The angles of the gradient in the upper and lower boundaries of the image.
2. The transition profile (the number of background-to-foreground transitions in a column).
3. The distance (in pixels) between the upper and the lower boundaries of the image.
4. The vertical projection profile (the number of foreground pixels within each column).

Each individual feature was normalized to the range \([0, 1]\), then the feature set per image column was extracted and combined to create a single multi-dimensional vector for all the columns in \(I\), as described in Ref. 14. For comparing two feature vectors, we use the DTW algorithm.

### 3.3 Gradient, Structural, Concavity features

Gradient, Structural, Concavity feature extraction approximates a multi resolution approach. Several distinct feature types are extracted at local, intermediate and large scales in the image. The local-scale features describe edge curvature in a neighborhood of a pixel, the intermediate features describe short stroke, and the large features describe concavities of the image object. In order to extract the GSC features a \(8 \times 4\) grid is placed on the image window using equimass pixel subdivisions in both vertical and horizontal directions. Then, a vector containing 1024 binary digits is computed (384 digits correspond to gradient, 384 correspond to structural and 256 correspond to concavity). For more details of GSC features we refer the reader to Ref. 16,17.

To measure the similarity between two binary vectors representing GSC features of two images, we use the similarity measure from Ref. 18, which is defined in Eq. (2), where \(X\) and \(Y\) are two binary vectors, \(s_{ij}\) represents the number of corresponding binary digits of \(X\) and \(Y\) that have value \(i\) and \(j\).

\[
d(X,Y) = \frac{1}{2}\left(1 - \frac{s_{11}s_{00} - s_{10}s_{01}}{(s_{10} + s_{11})(s_{01} + s_{00})(s_{11} + s_{01})(s_{00} + s_{10})} \right)
\] (2)

### 3.4 Histogram of oriented gradients

The histogram of oriented gradients (HOG)\(^{19}\) is widely used for object detection. HOG describes the shape by the distribution of edge directions. The implementation of the descriptor is achieved by decomposing an image into small cells of equal size. Then, histogram of oriented gradients in each cell is computed, and the combination of these histograms represents the descriptor.

In our implementation we first preprocess the candidate image patch by adding margins of equal size, and afterwards resizing it to a fixed size. The resulted patch is divided into cells, and the patch descriptor is obtained by the concatenation of HOGs of all cells. To compare two HOG descriptors we use the \(\chi^2\) distance, which is an appropriate measure for comparing two histograms. The \(\chi^2\) distance is defined as:

\[
\chi^2(H_1, H_2) = \frac{1}{2} \sum_{i=1}^{b} \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)}
\] (3)

where \(H_1\) and \(H_2\) are two histograms with \(b\) bins.
3.5 Local binary patterns

Local binary patterns (LBP) assign to each pixel in the image a binary number which is obtained by comparing the pixel value with its eight neighbors. The corresponding bit is assigned '1' if the pixel’s value is greater than the neighbor’s value, and '0' otherwise. This gives 8-digit binary numbers, which are usually further quantized. In order to obtain LBP descriptor of an image patch, we proceed similarly to Sec. 3.4. Margins are added to the patch image, afterwards it is resized to a fixed size and subdivided into cells. The histograms of the LBPs in each cell are concatenated to form the patch descriptor. We use $\chi^2$ distance from Eq. (3) to measure the similarities of two LBP descriptors, as the LBP descriptor is also built upon histograms.

4. EXPERIMENTS AND RESULTS

We tested our algorithm on Pinkassim† and Saint Gall datasets, sample pages of which are presented in Figure 3. Pinkasim (from Hebrew: minutes) were specially designated registers to record the activities of Jewish communities, such as the results of annual elections to the executive board and sub-committees, appointments of rabbis, marriage, death and burial registers, the expenses and the incomes, memory of historical events, and even communal conflicts. The documents we used in our tests are from the Pinkass of Frankfurt community, dated between 16th to 18th centuries. The pages are written by different authors in Yiddish using cursive Hebrew letters. The Saint Gall database was introduced in Ref. 6. It is a Latin manuscript containing the hagiography Vita sancti Galli by Walafrid Strabo. The document was written in the 9th century by a single writer.

To evaluate the algorithm we follow the evaluation protocol from Ref. 6, which calculates an alignment accuracy ($\text{Acc}$). The $\text{Acc}$ is defined in Eq. (4), where $N$ is the number of words in the ground truth, $S$ is the number of substitutions, $D$ is the number of deletions, and $I$ is the number of insertions. The values of $S$, $D$, and $I$ are calculated as string edit distance between the result of the alignment and the ground truth. Word boundaries are considered correct if they lie within the word spacing area, taking into account a tolerance of small number of pixels. Our algorithm assumes that the synthetic and original lines contain the exact word sequence, and as a consequence the number of result labels is equal to the number of words in the synthetic line. That is why, all the disparities between the ground truth and the results are counted as substitutions, and the values of $I$ and $D$ are always equal to zero. The evaluation in Ref. 6 also includes the calculation of precision and recall. These values are irrelevant to us due to the same reason described above.

$$\text{Acc} = \frac{N - S - D - I}{N} \quad (4)$$

The goal of the first experiment was to compare the performance of the algorithm with different features and image matching methods. This experiment was applied to six pages (130 lines) of the Pinkassim dataset. The results of this experiment are presented in Table 1a, and Figure 4 illustrates few examples of the alignment with GSC features. The top line in each example is the synthesized line, and the bottom is the corresponding original line. The aligned words in the two lines are shown with the same color. As we previously mentioned, the documents of the Pinkassim dataset are written by different authors, and as a consequence, the script in the synthetic lines is not always similar to the original script (e.g. Figure 4c). Nevertheless, the obtained results are promising. The alignment with profile-oriented features yielded the worst result, hereof we excluded this class of features from our next experiment.

In the second experiment we ran the algorithm on 40 pages (942 lines) of the Saint Gall dataset. Figure 5 illustrates some examples of the alignment. The results are compared to the results of Ref. 6 and are presented in Table 1b. The method presented in Ref. 6 uses HMM, and the published results are given for two experiments: the first uses one page for training (Ref. 6-(1)) and the second uses 20 pages for training (Ref. 6-(20)). The performance of our algorithm is better than Ref. 6-(1) and just a little worse than Ref. 6-(20). Given that our algorithm does not need any training, the obtained results demonstrate its potential. In addition, the expectation that both the synthetic and the original lines contain exactly the same word sequence (Sec. 3) can be relaxed by concatenating all the extracted text lines in the manuscript and considering it as one long line.

†http://www.yivoencyclopedia.org/article.aspx/Archives
Figure 3: (a) A page from the Pinkas of Frankfurt community; (b) A page from the Saint Gall manuscript.
Table 1: (a) Comparative performance of the algorithm with different feature extraction methods on the Pinkassim dataset; (b) Comparison of the results of our algorithm with the results presented in Ref. 6 on Saint Gall dataset. Ref. 6-(1) and Ref. 6-(20) are the results of the algorithm from Ref. 6 with one and with 20 pages for training respectively.

<table>
<thead>
<tr>
<th>Features used</th>
<th>Acc</th>
<th>Algorithm</th>
<th>Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile-based</td>
<td>58.63</td>
<td>Ours (with GSC)</td>
<td>88.14</td>
</tr>
<tr>
<td>GSC</td>
<td>71.10</td>
<td>Ours (with HOG)</td>
<td>88.40</td>
</tr>
<tr>
<td>HOG</td>
<td>73.89</td>
<td>Ours (with LBP)</td>
<td>89.17</td>
</tr>
<tr>
<td>LBP</td>
<td>65.67</td>
<td>Ref. 6-(1)</td>
<td>83.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ref. 6-(20)</td>
<td>92.07</td>
</tr>
</tbody>
</table>

Figure 4: (a)-(d) Four results of the alignment on the Pinkassim dataset. The top line in each example is the synthesized line, and the bottom is the corresponding original line. The aligned words in the two lines are get the same color.
Figure 5: (a)-(b) Two results of the alignment on Saint Gall dataset. The top line in each example is the synthesized line, and the bottom is the corresponding original line. The aligned words in the two lines are shown in the same color.

5. CONCLUSIONS AND FUTURE WORK

We presented a simple algorithm for aligning document images with their transcription. The algorithm does not require any training or recognition procedure, and is general in sense that it allows using various feature extraction and image distance functions. We have tested the algorithm on two datasets of historical documents, written in completely different languages, and for both datasets obtained promising results.

In the future work we plan to extend our algorithm in two directions. The first direction is to deal with an inaccurate transcription, where the manuscript and the transcription differ in small portions. The differences may appear as inserted, deleted or altered words in the transcription. Insertions and word altering may occur when abbreviations are written out in full, or when words are added by the transcriber to explain ideas. Deletions may occur when the word was crossed out (marked as deleted in the origin) and as a result was skipped out in the transcription. The second direction is to extend our algorithm to work directly on gray-scale images. Also, we plan to test the algorithm on more datasets of historical documents.

ACKNOWLEDGMENTS

This research was supported in part by the DFG-Trilateral grant no. FI 1494/3 – 2, the Ministry of Science and Technology of Israel, the Council of Higher Education of Israel, the Lynn and William Frankel Center for Computer Sciences and by the Paul Ivanier Center for Robotics and Production Management at Ben-Gurion University, Israel.

REFERENCES


