Transcript mapping for handwritten Chinese documents by integrating character recognition model and geometric context

Fei Yin*, Qiu-Feng Wang, Cheng-Lin Liu
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, 95 Zhongguan East Road, Beijing 100190, PR China

1. Introduction

Handwriting recognition research has been pursued for more than 40 years and many effective methods have been proposed [1–6]. For all the research works, standard datasets of characters and document images play crucial roles in system design and evaluation. Particularly, with the increasing use of statistical learning-based algorithms for handwritten document recognition and retrieval, large datasets of unconstrained handwritten documents become more demanding. However, to the best of our knowledge, no offline handwritten Chinese document datasets ground-truthed to text lines and characters have existed except the one released recently by the authors [7]. Therefore, ground-truthing document images, i.e., annotating the regions, text lines, words and characters, become a prerequisite for handwritten Chinese document analysis research. On the one hand, the design of segmentation and recognition algorithms needs a large number of labeled data for training. On the other hand, the performance of segmentation and recognition should be evaluated and refined on labeled data. By ground-truthing, the text regions, text lines and words/characters need to be segmented and tagged very accurately. Unfortunately, such ground-truth was typically created manually and therefore its creation is tedious, expensive and prone to human errors. To a large extent, automatic annotation tools can alleviate above difficulties.

Aligning text line images with their text transcripts is the crucial step of handwriting annotation. However, even with the transcript available (especially for character level alignment), the alignment is not a trivial problem. First of all, the characters cannot be reliably segmented because the size and position of characters are variable and the strokes are often touching and overlapping in handwritten documents. For unconstrained Chinese handwritten documents, there is no extra gap between words than characters (Fig. 1 shows an example of Chinese document image and its character level transcript mapping). Automatic segmentation may result in errors of over-segmentation (one character segmented into two or more segments) and under-segmentation (two or more characters merged into one segment), thus the total number of segmented character images usually is not equal to the total number of characters in the transcript. Therefore, a simple linear alignment does not work.

In recent years, many efforts have been devoted to this difficult problem, and some ground-truthing tools have been developed for automatic annotation of document images generated with less restriction [8–30]. For printed document images synthesized
by text editing, printing and scanning, the text description (transcript) can be matched with the scanned image to generate ground-truth data automatically [8–13]. Handwritten document images can be similarly matched with the transcript (existing a priori or edited a posteriori), but the matching process is much more complicated due to the irregularity of document layout and written word/character shapes [14]. Usually, for handwritten documents, a dynamic search algorithm optimizing a match score between the text line image and its transcript is used to align the sequence of image segments and the words/characters. These methods can be roughly categorized into two groups depending on whether word/character recognition models are used or not. In the first group without recognition models, outline features are extracted from text line image for transcript alignment [15–23]. Recognition models can better measure the similarity between image segment and word/character and thus can improve the alignment accuracy. The methods in [24–26] use character prototype-based word models, the ones in [27–30] use HMM-based word recognizers. Taking into account the salient effects of character recognizers in word/string recognition, the recognizer largely benefits the alignment of text lines.

The alignment accuracy is still not sufficient despite the promise of recognition models. In Chinese documents, the mixed alphanumeric characters and punctuation marks are prone to segmentation and labeling errors because they have distinct geometric features of size, aspect ratio and position in text line. The misalignment of Chinese characters is mainly due to character touching and the gaps within characters composed of multiple radicals. Fig. 2 shows typical annotation errors caused by a punctuation mark and a radical of Chinese character (green box). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this article.)

2. Related works

There have been many efforts in developing methods and systems for minimizing the human labors in document images ground-truthing. The published methods can be roughly categorized into three classes: geometric transformation-based, word matching-based and recognition-based methods.

Geometric transformation-based methods are mainly designed for printed document alignment. The underlying idea is to estimate a global geometric transformation and then perform a robust local bitmap match between the text description image (generated using a word-processing tool) and document image (printed, photocopied, or scanned from text description). For such matching, Kanungo and Haralick [8,9], Zi and Doermann [10] used four corner points of the images as corresponding pairs to find a linear transformation. These methods are not robust when an image part is missing or there are extra feature points. Hobby [11] improved this method by considering all feature points and using a direct search optimization algorithm [38] to search for the affine transform parameters by minimizing the matching cost between the text description and the character bounding boxes in the image. This method, however, suffers from heavy computational burden and local minimum. Kim and Kanungo [12,13] improved the method by grouping the feature points to reduce the complexity and employing a modified attributed branch-and-bound matching algorithm to guarantee global minimum. Though these methods do not consider the deformation of layout and character shapes, Viard-Gaudin [14] et al. tried this methodology to create ground truth for handwritten documents. They designed a database of online and offline handwritten data by...
manually locating corresponding points in online and offline domain, but the matching process is much complex.

Annotating handwritten document images is usually addressed by formulating the matching between text line image and its transcription as an alignment of two ordered sequences. Depending on whether word/character recognition models are used or not, these methods also can be grouped into word matching-based methods and recognition-based methods.

In word matching-based methods, Zinger et al. [15] noticed that the relative lengths of ASCII and handwritten words are highly correlated. They proposed to find the word level alignment by sequentially adjusting the word image boundaries from right to left with the cost function based on relative length. Stamatopoulos et al. [16] adopted a greedy search algorithm which recursively separates the text line according to the between-component gaps until the word number equals that of the transcription. In the approach of Kornfield et al. [17,18], the document was automatically segmented into a list of word images, which were matched with text using DTW based on global geometric properties (word-box features) extracted from both the handwritten word image and the transcription with a special font. Except using different matching features and replacing transcript with synthesized handwritten image based on writer’s handwriting model in [21,22], the DTW was similarly employed in [19–22]. The hidden Markov model (HMM) is another effective tool to solve the alignment problem. It was used in the work of Rothfeder et al. [23], where the word images were treated as the hidden variables, and the HMM models the probability of observing the word image given the word text. The Viterbi algorithm was used to decode the sequence of word images.

Among the recognition-based methods, the one of Tomai and Zhang et al. [24,25] formulates word alignment as an optimization problem involving multiple word segmentation hypotheses and word recognition, and uses dynamic programming to find word alignment in several coarse-to-fine stages. The word recognizer provides lexicon words as well as the character boundaries aligned with a word. The extended work of Huang and Srihari [26] used a similar approach. Zimmermann and Bunke [27] proposed an automatic word segmentation method with HMM-based word recognizer for cursive handwritten text lines alignment, in which the word HMM was concatenated by character models, and the word models were concatenated to align with the transcription of text line. It worked very well on the IAM off-line datasets. A similar method was employed in [28]. Liwicki et al. [29] proposed to improve the performance by combining the alignment results given by multiple different HMM-based word recognizers. Recently, Toselli et al. [30] also combined the human transcriber and a HMM based handwritten text recognition system to generate the transcription of text images.

Our proposed approach uses a character recognizer for handwritten Chinese text lines alignment. Character-level alignment by DTW is suitable for Chinese documents because of the large category set (over 5000 characters are frequently used) and the rich shape information of single characters. We also utilize geometric context information to improve the alignment accuracy. HMM is an effective model for text line alignment and has been widely used to annotate the western text document databases. However, for handwritten Chinese text recognition, HMM has not shown superior performance (e.g., only 39% character correct rate was reported in [40]). The proposed DTW-based algorithm is simpler than HMM in implementation. On the other hand, the state sequence decoding process in HMM is similar to DTW, both follow the principle of dynamic programming.

3. System overview

The block diagram of our Chinese handwritten document annotation system is shown in Fig. 3. The input to the system is a handwritten document image and its corresponding transcript (examples given in Fig. 4), the output is the ground-truth data (annotated image). First, the number of text lines and the number of characters in every line are detected from the transcript of the document. The document image is segmented into text lines guided by given number, and then each text line image is aligned with its transcript for character segmentation and labeling. In our practical tool, line segmentation and alignment are automatic and the remaining errors can be corrected by human intervention. In this paper, however, we focus on the automatic alignment and the manual corrections are not involved in the performance evaluation.

Separation of text lines in handwritten documents is not a trivial task because the text lines are often un-uniformly skewed and curved, and the inter-line space is not prominent. We have designed a text line segmentation algorithm based on minimal spanning tree (MST) clustering with distance metric learning [39], in which the connected components (CCs) are grouped by MST into a tree structure under a learned metric, and then the edges of

![Fig. 3. Block diagram of the annotation system.](image)

![Fig. 4. An example of input document image and its corresponding transcription. (a) Input document image. (b) Transcription for the document image.](image)
the tree are dynamically cut by using a new hypervolume reduction objective to get the final text lines. This algorithm performs very well, but is not perfect. For example, sometimes a line is split into several sub-lines or several lines are merged into one. Therefore, we designed a series of post-processing operations to correct mis-segmented text lines manually. The design of human corrections is trivial and thus is not detailed in this paper.

After separating the document image into the same number of text lines as the text transcript, each text line image is to be segmented into characters and aligned with its corresponding transcript. Since the characters cannot be reliably segmented before recognition, we solve it using the over-segmentation strategy, which can take advantage of the character shape, overlapping and touching characteristics to better separate the characters at their boundaries. The result of over-segmentation is trivial and thus is not detailed in this paper.

After over-segmentation, the ordered sequence of primitive segments of each text line is dynamically partitioned into character patterns by matching with the character recognition model. The optimal match can be found by dynamic programming (DP) search (also called dynamic time warping (DTW) or Viterbi algorithm in special contexts) to minimize an edit distance, and the alignment result largely depends on the edit costs defined for segment-to-character match. The character recognizer measures the shape similarity (or distance) between a candidate pattern and from the pair of the (composed of one or more consecutive primitive segments) and a character, which is desired to assign high score to correct character classes and low scores to incorrect classes. On a non-character pattern, the classification score is desired to be low for all character classes [42]. We use geometric models to measure the similarity of character outline and between-character compatibility. The best alignment, corresponding to a path in a grid space, is searched for by DTW. Fig. 6 shows an example, where a string of seven characters is aligned with 10 primitive segments, which are correctly segmented and labeled by the path of the red line. After alignment, mis-segmentation and mis-labeling of characters (such as the path of blue line in Fig. 6, and such errors are inevitable) can also be corrected manually in post-processing [36].

4. Text line alignment

Text line alignment is the crucial step of document annotation. In this section, we first give a statistical formulation of this problem and a DTW-based solution, then describe the techniques for the involved confidence transformation (CT) and parameters optimization.

4.1. Problem formulation

We formulate the problem of text line alignment from Bayesian decision view so as to integrate geometric context and character recognition model in a principled way. Under the 0/1 loss, the optimal criterion for alignment is to maximize the posterior probability of the character segmentation given the text line image (X) and its transcript containing n characters \((T = t_1...t_n)\). This posterior probability can be formulated by

\[
P(s|X,T) = \frac{P(s,X|T)}{P(T)} = \frac{P(s,X)p(T|X,s)}{P(T)}
\]

\[
= \frac{P(s|X)p(T|X)}{P(T)} = P(s|X)p(T|X)
\]

(1)

where \(X\) denotes the sequence of candidate characters corresponding to segmentation \(s\). Hence, the optimal segmentation is decided by

\[
s^* = \arg \max_{|s| = |T|} P(s|X)p(X|T)
\]

(2)

where the segmentation is also constrained to have the same length as the transcript \(|s| = |T|\).

In formulation (2), \(P(s|X)p(X|T)\) denotes the posterior probability of the \(s\)-th segmentation path given the text line image \(X\). It can be decomposed into

\[
P(s|X) = \prod_{i=1}^{n}p(z_i^p = 1|g_i^p)p(z_i^f = 1|g_i^f)
\]

(3)

where \(n\) is the character number of transcript, \(z_i^p = 1\) indicates that the \(i\)-th candidate pattern is a valid character, and \(z_i^f = 1\) indicates that the gap between the \((i-1)\)-th and \(i\)-th candidate patterns is a valid between-character gap; \(g_i^p\) and \(g_i^f\) denote the class-independent geometric features extracted from the \(i\)-th candidate pattern, and from the pair of the \((i-1)\)-th and \(i\)-th candidate patterns, respectively. The two probabilistic terms in (3) correspond to the unary and binary class-independent geometric models (see Section 5.2).

The likelihood function \(P(X|T)\) can be decomposed into the product of character-dependent terms since the segmentation \(X\) is a sequence of candidate character patterns

\[
P(X|T) = \prod_{i=1}^{n}p(x_i|t_i)p(g_i^p|t_i)p(g_i^f|t_{i-1}t_i)
\]

(4)

...
where $x_i$, $g_i^{mc}$ and $g_i^{bc}$ denote the shape features used in character recognition, unary and binary geometric features (see Section 5.1), respectively. Similarly, we can decompose $p(X)$ as

$$p(X) = \prod_{i=1}^{n} p(x_i) p(g_i^{mc}) p(g_i^{bc}).$$  

(5)

Combining (4) and (5), $P(X|T)/P(X)$ is obtained by

$$P(X|T)/P(X) = \prod_{i=1}^{n} \frac{p(x_i|t_i) p(g_i^{mc}|t_i) p(g_i^{bc}|t_i-t_{i-1})}{p(x_i) p(g_i^{mc}) p(g_i^{bc})}.  

(6)$$

Since the probabilities $p(x_i|t_i)$, $p(g_i^{mc}|t_i)$ and $p(g_i^{bc}|t)$ are not trivial to estimate, we convert (6) (7),

$$P(X|T)/P(X) = \prod_{i=1}^{n} \frac{p(t_i|x_i) p(x_i|t_i) p(g_i^{mc}|t_i) p(g_i^{bc}|t_i-t_{i-1})}{p(x_i) p(g_i^{mc}) p(g_i^{bc})}.  

(7)$$

where the three posterior probabilities can be approximated by confidence transformation of classifier outputs (see Section 4.3), and the three corresponding prior probabilities $p(t_i|p_i)$ and $p(t_i-t_{i-1})$ are viewed as constants in classifier design, denoted by $p_1$, $p_2$, $p_3$, respectively ($p_1$ and $p_2$ are different due to character class clustering in geometric model, see Section 5.3).

Combining (3) and (7), the optimal segmentation of (2) is given by

$$s^* = \arg \max_{\{s\} = \{t\}} \frac{1}{n} \prod_{i=1}^{n} \left( p(t_i|x_i)p(t_i|x_i) p(g_i^{mc}|t_i) p(g_i^{bc}|t_i-t_{i-1}) \right).  

(8)$$

where $P = p_1, p_2, p_3$. Though the formulation (8) approximates the posterior probability of optimal segmentation fairly well, it is still insufficient because the geometric models and character recognition model do not always meet the assumptions. To consider the effects of different models and achieve a better performance, we take the logarithm of probability and incorporate the weights of different models to get a generalized likelihood function $f(X, T)$ for segmentation path evaluation

$$f(X, T) = 2\alpha \sum_{i=1}^{n} \log p(t_i|x_i) + \lambda_1 \sum_{i=1}^{n} \log p(t_i|x_i) + \lambda_2 \sum_{i=1}^{n} \log p(t_i|s_i) + \lambda_3 \sum_{i=1}^{n} \log p(t_i|s_i) + \lambda_4 \sum_{i=1}^{n} \log p(t_i|s_i)$$

$$= \sum_{i=1}^{n} \log p(t_i|x_i) + \lambda_1 \sum_{i=1}^{n} \log p(t_i|x_i) + \lambda_2 \sum_{i=1}^{n} \log p(t_i|s_i) + \lambda_3 \sum_{i=1}^{n} \log p(t_i|s_i) + \lambda_4 \sum_{i=1}^{n} \log p(t_i|s_i),  

(9)$$

where the term $\sum_{i=1}^{n} \log 1/P$ has been omitted because it is a constant for all segmentation paths given $T$. Hence, the optimal segmentation can be defined as

$$s^* = \arg \max_{\{s\} = \{t\}} f(X, T).  

(10)$$

4.2. String alignment with dynamic time warping

After over-segmentation, a text line image is represented as a sequence of primitive segments ordered from left to right $X = [x_1, x_2, \ldots, x_m]$,  

(11)

where $x_j$ is the $j$-th segment and $m$ is the number of primitive segments in this line. A primitive segment is a character or a part of a character and the concatenation of adjacent segments form candidate character patterns. The transcript $T$ is a character string $T = [t_1, t_2, \ldots, t_n]$,  

(12)

where $t_j$ is the $j$-th character and $n$ is the number of characters in the transcript ($n \leq m$). An example of text line image and its text transcript is shown in Fig. 7, where seven characters are to be aligned with 10 segments.

A possible mapping $X'$ between a text line image $X$ and its corresponding transcript $T$ is defined as

$$X' = [(t_1, x_1), \ldots, (t_n, x_n), \ldots, (t_n, x_n)] = [(t_1, x_1, \ldots, x_{k-1}, h, \ldots, t_1, x_{j-k+1}, \ldots, x_j), \ldots].$$

where $x_{j-k+1}, \ldots, x_j$ is an ordered sequence of primitive segments, which are concatenated into a character pattern $x_i^0$ to match with character $t_k$. $k$ is the number of primitive segments that form the candidate character pattern for $t_k$.

The definition of alignment in (13) corresponds to a path from bottom left to top right in the grid of Fig. 6. The alignment cost in (9) can be rewritten as

$$f(X, T) = \sum_{h=0}^{4} \lambda_h \cdot F_h,  

(14)$$

where $\lambda_h$ ($h = 0, \ldots, 4$) are the weight coefficients, $F_0$ is the character recognition score, $F_1, F_2, F_3$ and $F_4$ are four geometric model scores, respectively. Each term is the sum of scores over the segmentation path

$$F_0 = \sum_{i=1}^{n} \log p(t_i|x_i) = \sum_{i=1}^{n} f_0(t_i|x_{k_j-1,j-1} \ldots),  

(15)$$

$$F_1 = \sum_{i=1}^{n} \log p(t_i|x_i) = \sum_{i=1}^{n} f_1(t_i|x_{k_j-1,j-1} \ldots),  

(16)$$

$$F_2 = \sum_{i=1}^{n} \log p(t_i|x_i) = \sum_{i=1}^{n} f_2(t_i|x_{k_j-1,j-1} \ldots),  

(17)$$

$$F_3 = \sum_{i=1}^{n} \log p(t_i|x_i) = \sum_{i=1}^{n} f_3(x_{k_j-1,j-1} \ldots),  

(18)$$

$$F_4 = \sum_{i=1}^{n} \log p(t_i|x_i) = \sum_{i=1}^{n} f_4(x_{k_j-1,j-1} \ldots),  

(19)$$

where $f_0(\cdot), f_1(\cdot), f_2(\cdot), f_3(\cdot), f_4(\cdot)$ are the logarithms of confidence of character recognition, class-dependent single character and between-character geometry, class-independent single character and between-character geometry, respectively. Note that the class-independent between-character geometry ($f_0$) considers two adjacent primitive segments only, while the class-dependent between-character geometry ($f_2$) considers two adjacent candidate characters.

As the alignment cost is the summation of multiple stages, the optimization problem (10) can be solved by searching for the minimum negative cost path $\langle f(X, T), DTW \rangle$ with DTW. To do this, we define $D(i, j)$ as the accumulated cost of optimal alignment between a partial string $[t_1 \ldots t_i]$ and partial image $[x_1 \ldots x_j]$. $D(i, j)$ can be updated from the preceding partial alignments by

$$D(i, j) = \min_{k_i} \left\{ D(i, j-k) + \text{penalty}(x_{k_j-1,1} \ldots x_j) + \varphi, \right.$$

$$D(i-1, j) + \text{penalty}(t_i),$$

$$\left. D(i-1, j-k) + \theta, \right\}$$

(20)
\[ \theta = \lambda_0 \phi_0(t_j|X_{j-k+1},...X_j) + \lambda_1 \phi_1(t_j|g^{x_{k,h+1}},...,x_{k,h+1}) \\
+ \lambda_2 \phi_2(t_j^+,|g^{x_{k+1},h+1},...,x_{k+1}) \\
+ \lambda_3 \phi_3(z_j^+ = 1|g^{x_{k+1},h+1},...,x_{k+1}) + \lambda_4 \phi_4(z_j^+ = 1|g^{x_{k+1},h+1},...,x_{k+1}), \]  

(22)

where \( t_j^+ \) represents the preceding non-skipped character of \( t_i \). That is, \( t_j^+ = t_i \) if \( |\{i\}| \) is not skipped, otherwise recursively set \( l = l-1 \) until \( t_i \) is not skipped.

The DTW search starts with \( D(0,0) = 0 \), then for \( i = 1,...,n \), and \( j = 1,...,m \), \( D(i,j) \) are iteratively updated according to (20) and the optimal number \( k_i \) (number of primitive segments concatenated) is stored for \( \{i\} \). Finally, \( D(m,n) \) gives the total cost of optimal alignment. According to our over-segmentation technique, we allow a candidate pattern to be formed by at most four primitive segments (namely, \( 1 \leq k_i \leq 4 \)).

After we get the optimal alignment, the partition of primitive segments can be retrieved by backtracking \( k_i \) from \( (m,n) \) to the start. This partition gives the segmentation of text line image into characters.

4.3. Confidence transformation

For probabilistic fusion of classifier outputs, the transformed confidence measures are desired to approximate the class posterior probability \( p(o|x) \) (\( o \) refers to a class and \( x \) is the feature vector). Though the posterior probability can be directly obtained by the Bayes formula given a priori probability and the conditional probability density of each class, the probability density functions are not trivial to estimate. Instead, there are many ways to approximate the posterior probability from classifier outputs, such as sigmoidal function, soft-max function and Dempster–Shafer theory of evidence [43]. The formulation (9) shows that the posterior probabilities indicate whether a candidate character or geometric feature belongs to a specific class or not. Therefore, the sigmoidal function (23), which is often taken for binary posterior probability, is a good choice for our problem

\[ p(o_j|x) = \frac{\exp[-a d_j(x) + \beta]}{1 + \exp[-a d_j(x) + \beta]}, \quad j = 1,...,M, \]  

(23)

where \( M \) is the number of defined classes considered by the classifier, \( d_j(x) \) is the dissimilarity score of class \( o_j \), \( a \) and \( \beta \) are the confidence parameters.

We optimize the confidence parameters by minimizing the cross entropy (CE) loss function (24), which is commonly used in logistic regression and neural network training. On a validation dataset (preferably different from the dataset for training the classifier) of \( N \) samples, the CE is

\[ \min CE = -\frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{M} [t_{n,j} \log P_j + (1-t_{n,j}) \log (1-P_j)], \]  

(24)

where \( P_j = p(o_j|x) \), \( t_{n,j} = \delta(c^e_j, e)/\{0,1\} \) and \( c^e \in \{1,2,...,M\} \) is the class label of the \( n \)-th sample. The CE is minimized by stochastic gradient descent to update the confidence parameters.

4.4. Parameters optimization

The objective of training is to tune the combining weights of (9) and the penalty parameters of (20) to optimize the alignment performance. However, there exist two types of trained parameters in our alignment model. The differentiable combining weights \( \{\lambda_0,...,\lambda_4\} \) can be optimized by stochastic gradient ascent; the penalty parameters \( \{\text{penalty}(t_i),\text{penalty}(X_{j-k+1},...X_j)\} \) are not differentiable and hence do not allow optimization by gradient search. Fortunately, we can optimize them with direct search method which has been used in [44] to search for optimal parameters for page segmentation algorithm. Therefore, we design a two-stage optimization strategy. In the first stage, we disable the deleting and skipping operations by fixing the penalty parameters to a maximum, and train the combining weights on a training set which contains neither deleted primitive segments nor skipped characters. In the second stage, the penalty parameters are optimized by fixing the combining weights on another training set containing all types of text line samples.

4.4.1. String-level MCE training for weight parameters

The aim of this training stage is to tune the combining weights so as to promote correct alignment and depress incorrect alignment. String-level training with the minimum classification error (MCE) criterion has been widely used in speech recognition and handwriting recognition and has reported superior performance [45,46].

In string-level training, the weights are estimated on a dataset of string samples \((X^n, S^n)| n = 1,...,N\), where \( S^n \) denotes the correct alignment (i.e., correct segmentation) of the sample \( X^n = (X^t, T^n) \), by optimizing an objective function related to text line alignment performance. Denoting the cost of a sample \( X^n \) and its alignment \( S^n \) as \( g(X^n, S^n, \Lambda) \), where \( \Lambda = \{\lambda_0,...,\lambda_4\} \) is the set of weights. In string-level MCE training described in [47], the misclassification measure for correct alignment \( S^n \) is approximated by

\[ d(X^n, S^n, \Lambda) = -g(X^n, S^n, \Lambda) + g(X^n, S^n, \Lambda). \]  

(25)

where \( g(X^n, S^n, \Lambda) \) is the cost of the closest rival (the minimum cost alignment excluding the correct one)

\[ g(X^n, S^n, \Lambda) = \max_{k \neq \text{ref}} g(X^n, S^n, \Lambda). \]  

(26)

The misclassification measure is transformed to loss by the sigmoidal function

\[ L(\Lambda) = \frac{1}{1 + e^{-b d(\Lambda,\Lambda)}}, \]  

(27)

where \( b \) is a parameter to control the hardness of sigmoidal nonlinearity. On the training sample set, the empirical loss is regularized to overcome the ill-posedness

\[ L(\Lambda) = \frac{1}{N} \sum_{n=1}^{N} L(X^n, \Lambda) + \frac{\beta}{2} \|\Lambda\|^2. \]  

(28)

By stochastic gradient descent, the parameters are updated on each training sample by

\[ \Lambda(t+1) = \Lambda(t) - \epsilon(t) \frac{\partial L(X^n, \Lambda)}{\partial \Lambda} |_{\Lambda = \Lambda(t)} \]  

(29)

where \( \epsilon(t) \) is the learning step. Accordingly, the updating formula of each weight parameter can be derived as

\[ \lambda_{h+1}^t = (1-\beta) \lambda_h^t - \epsilon(t) \left( 1 - h \right) \sum_{i=1}^{L} \left( f'_h(t_i, S_i^0) - f_h(t_i, S_i^0) \right). \]  

(30)

where \( L \) is the string length of sample \( X^n \) (the number of characters in the text transcript), \( h = 0,...,4 \), and \( (t_i, S_i^0) \) denotes the alignment of character \( t_i \).

In implementation, the weights are initialized as equal values and then iteratively updated on each string sample based on (30). And by retaining several alternative alignments in dynamic programming, we obtain the rival alignment which is most confusable with the correct one.

4.4.2. Simplex search for penalty parameters

Though the penalty parameters have important impact on the alignment performance, they are typically manually selected and no training method is explicitly sought [18,19,22,26]. Fortunately, this optimization problem is a multivariate nonsmooth nonlinear function optimization problem as those in [44] and can be solved
using the simplex method. We use a public domain implementa-
tion of downhill simplex method proposed by Nelder and Mead [38], and use the standard choice for reflection (1), contraction (0.5), expansion (2), shrinkage (0.5) and stopping threshold (10−6) coefficients. The downhill simplex method is a local optimization algorithm. Thus, for each (different) starting point, the optimization algorithm could converge to a different optimal solution. To overcome the local optimum, we constrain the parameter values to lie within a reasonable range and randomly choose some starting locations within this range. The solution corresponding to the lowest optimal value is chosen as the best optimal set of parameters.

5. Geometric context modeling

Since the distinct outline features of alphanumeric characters and punctuation marks can be exploited to improve the alignment accuracy, we design four statistical models for the class-dependent and class-independent single-character geometry (unary geometric context), and the class-dependent and class-independent between-character relationships (binary geometric context), respectively. In the following, we first describe the geometric features, and then describe the statistical models scoring these features.

5.1. Class-dependent geometric features

For modeling single-character geometry, we first estimate the average height of the text line image as in [41] because some geometric features are necessarily normalized w.r.t. to the height (i.e., divided by it) so as to be invariant to text line height. We extract 42 geometric features from a candidate character pattern, which are grouped into three categories: (1) 10 scalar features related to the bounding box of the character, as the No. 1–10 in Table 1. (2) Four scalar features related to the vertical center of text line, as No. 11–14 in Table 1. (3) 28 profile-based features inspired by the methods of [48,49], as No. 15–42 in Table 1.

We also extract 24 features for binary class-dependent geometric context, which are grouped into two categories: (1) 16 scalar features between the bounding boxes of two consecutive character patterns, as No. 1–16 in Table 2. (2) Eight features between the profiles of two consecutive character patterns, as No. 17–24 in Table 2.

5.2. Class-independent geometric features

To measure whether a candidate pattern is a valid character or not, we extract 12 class-independent geometric features. They are the same features of the No. 1–10 and the No. 15–16 in Table 1. The class-independent geometry between two consecutive candidate patterns describes whether an over-segmentation gap is a valid between-character gap or not. For this we extract 14 class-independent features from two adjacent primitive segments, 13 of them are the same as the No. 1–13 in Table 2, and the last is the convex hull-based distance as calculated in [50].

5.3. Statistical models

For modeling the class-dependent geometry, we should first reduce the number of character geometry classes, since the number of Chinese characters is very large and many different characters have similar outline geometry. Particularly, the binary class-dependent geometric model considers pairs of characters, and it is formidable to store M × M models (M is the number of character classes) and get enough training samples for so many models. Hence, we cluster the character classes into six superclasses (based on our prior knowledge of character shapes) using

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–2</td>
<td>Height and width of bounding box</td>
<td>Y</td>
</tr>
<tr>
<td>3</td>
<td>Sum of inner gaps</td>
<td>Y</td>
</tr>
<tr>
<td>4–5</td>
<td>Distances of horizontal/vertical gravity center to left/upper bound</td>
<td>Y</td>
</tr>
<tr>
<td>6</td>
<td>Logarithm of aspect ratio</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>Square root of bounding box area</td>
<td>Y</td>
</tr>
<tr>
<td>8</td>
<td>Diagonal length of bounding box</td>
<td>Y</td>
</tr>
<tr>
<td>9–10</td>
<td>Distances of horizontal/vertical gravity center to horizontal/vertical geometric center</td>
<td>Y</td>
</tr>
<tr>
<td>11–12</td>
<td>Distances of vertical gravity/geometric center to text line vertical gravity/geometric center</td>
<td>Y</td>
</tr>
<tr>
<td>13–14</td>
<td>Distances of upper/lower bound to text line vertical geometric center</td>
<td>Y</td>
</tr>
<tr>
<td>15–16</td>
<td>Means of horizontal/vertical projection profiles</td>
<td>Y</td>
</tr>
<tr>
<td>17–22</td>
<td>Normalized amplitude deviations, coefficients of skewness and kurtosis of the horizontal and vertical projection profiles</td>
<td>N</td>
</tr>
<tr>
<td>23–30</td>
<td>Means and deviations of the upper, lower, left and right outline profiles</td>
<td>Y</td>
</tr>
<tr>
<td>31–42</td>
<td>Normalized amplitude deviations, coefficients of skewness and kurtosis of the upper, lower, left and right outline profiles</td>
<td>N</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No.</th>
<th>Feature</th>
<th>Norm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–6</td>
<td>Distances between the upper bounds, lower bounds, upper-lower bounds, lower-upper bounds, left bounds and right bounds</td>
<td>Y</td>
</tr>
<tr>
<td>7–8</td>
<td>Distances between the horizontal gravity centers and the vertical gravity centers</td>
<td>Y</td>
</tr>
<tr>
<td>9–10</td>
<td>Distances between the horizontal geometric centers and the vertical geometric centers</td>
<td>Y</td>
</tr>
<tr>
<td>11–12</td>
<td>Height and width of the box enclosing two consecutive characters</td>
<td>Y</td>
</tr>
<tr>
<td>13</td>
<td>Gap between the bounding boxes</td>
<td>Y</td>
</tr>
<tr>
<td>14</td>
<td>Ratio of heights of the bounding boxes</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>Ratio of widths of the bounding boxes</td>
<td>N</td>
</tr>
<tr>
<td>16</td>
<td>Square root of the common area of the bounding boxes</td>
<td>Y</td>
</tr>
<tr>
<td>17–20</td>
<td>Differences between the mean of upper, lower, left and right outline profiles</td>
<td>Y</td>
</tr>
<tr>
<td>21–24</td>
<td>Differences between the deviation of upper, lower, left and right outline profiles</td>
<td>Y</td>
</tr>
</tbody>
</table>
the EM algorithm. After clustering, each single character is assigned to one of six super-classes and a pair of successive characters thus belongs to one of 36 binary super-classes. For estimating the statistical geometric models, training character samples are relabeled to six super-classes. We use a quadratic discriminant function (QDF) for both the unary and binary class-dependent geometric models. For unary class-dependent geometry, the 42-dimensional feature vector is reduced to 5-dimensional subspace by Fisher linear discriminant analysis (FLDA), and the projected samples are used to estimate the parameters of six-class QDF. For binary class-dependent geometry, the 36-class QDF is estimated using samples of 24-dimensional geometric features. The unary class-independent geometry indicates whether a candidate pattern is a valid character or not. For this two-class problem, we use a linear support vector machine (SVM) trained with character and non-character samples. The class-independent binary geometry indicates whether a segmentation point between two adjacent primitive segments is a between-character gap or not. We similarly use a linear SVM for this two-class problem, trained with two-class labeled samples.

The four geometric models in our system are summarized in Table 3.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unary class-dependent</td>
<td>42 → 5 QDF</td>
</tr>
<tr>
<td>Binary class-dependent</td>
<td>24 QDF</td>
</tr>
<tr>
<td>Unary class-independent</td>
<td>12 SVM</td>
</tr>
<tr>
<td>Binary class-independent</td>
<td>14 SVM</td>
</tr>
</tbody>
</table>

6. Experimental results

We evaluated the performance of our approach on a large database of unconstrained Chinese handwriting, CASIA-HWDB [7], and on a small dataset HIT-MW [51]. Because we focused on the performance of the automatic alignment algorithm, the following experiments did not involve any manual correction steps.

6.1. Database and experimental setting

The CASIA-HWDB database contains both isolated characters and unconstrained handwritten texts, and is divided into a training set of 816 writers and a test set of 204 writers. The training set contains 3,118,477 isolated character samples of 7356 classes (7185 Chinese characters, 109 frequently used symbols, 10 digits, and 52 English letters) and 4076 pages of handwritten texts. The test pages have a few mis-written characters and characters beyond the 7356 classes, which we call non-characters and outlier characters, respectively. We evaluated the text line alignment performance on 1015 handwritten pages of 204 test writers, which were segmented into 10,449 text lines containing 268,628 characters (including 723 non-characters and 368 outlier characters).

To validate the effectiveness of our alignment algorithm, we also tested on the dataset HIT-MW, from which a test set of 386 text lines contains 8448 characters (7405 Chinese characters, 780 symbols, 230 digits, 8 English letters, 16 non-characters and 9 outlier characters).

Note that there are under-segmentations errors after over-segmentation [41] and the characters of under-segmentation obviously cannot be aligned correctly. To evaluate the alignment performance more fairly, we also selected the text lines without under-segmentation errors. There are 8462 such lines in CASIA-HWDB and 304 lines in HIT-MW.

To build the character classifier, we first converted the grayscale character images to binary images, then extracted features using the continuous NCFE (normalization-cooperated feature extraction) method combined with the MCBA (modified centroid boundary alignment) normalization method [52]. The resulting 512-dimensional feature vector is projected onto a 160-dimensional subspace learned by Fisher linear discriminant analysis (FLDA), and then the 160-dimensional vector is input into a classifier. The classifier parameters were learned on the training samples of isolated characters in CASIA-HWDB.

For estimating the parameters of the geometric models, we extracted geometric features from 41,781 text lines of training text pages.

For training the alignment model, we first filtered the training text lines by removing the strings with under-segmentation errors (cases that correct segmentation points are not included in the candidate search grid), then selected 5000 text lines which contain neither deleted primitive segment nor skipped character to train the weight parameters; at last, the penalty parameters were learned on 500 text lines (in which 250 text lines contain deleted primitive segments, the others were constructed by inserting skipped characters randomly) selected from the remaining samples.

Though the annotation system (Fig. 3) involves text line separation, we focus on the performance of text line alignment (character segmentation and labeling), whereas the performance of text line separation has been evaluated in [39].

Using a state-of-the-art over-segmentation algorithm [41] on the test pages of CASIA-HWDB database, we observed that 4.46% of characters were not correctly separated (a character is correctly over-segmented when it is separated from other characters despite the within-character splits), i.e., they were under-segmented. These characters cannot be correctly segmented and aligned by combining consecutive primitive segments. This implies that the over-segmentation of characters is still a challenge. Some examples of under-segmentation errors are shown in Fig. 8.

Our system was implemented on a PC with Intel Quad Core 2.83 GHz CPU and 4 GB-RAM, and the algorithms were programmed in MS Visual C++ 2008.

6.2. Performance metrics

We evaluate the alignment performance of text lines using an accuracy metric based on bounding boxes of characters [24–27].
which is called alignment rate (AR)

\[ AR = \frac{N_c}{N_t} \]

where \( N_c \) is the number of correctly aligned characters and \( N_t \) is the total number of characters in the transcript.

In our experiments, a match between a transcript character and primitive segments, \((t_i, x_{k_i+1}, \ldots, x_{l_i})\), is judged as correct if the bounding box of the primitive segments and the bounding box of the true character image overlap sufficiently (the difference of top, bottom, left and right bounds do not exceed 1.5 times of the estimated stroke width of the test text line).

### 6.3. Text line alignment performance

We first evaluate the performance of confidence transformation and geometric models on the test text line data of CASIA-HWDB and HIT-MW; then discuss the effects of different geometric models; third, we compare our recognition-based method with word matching-based methods; and last, the alignment errors are analyzed in details.

In addition, to evaluate the effects of different character classifiers combined with geometric models, we tested two types of classifiers: MQDF (modified quadratic discriminant function) [53] classifier and NPC (nearest prototype classifier), which have been widely used in handwritten character recognition as well as handwritten document retrieval [54]. The parameters of MQDF classifier are estimated by maximum likelihood estimation of class means and covariance matrices, eigenvalue decomposition of covariance matrices, and replacing the minor eigenvalues with a constant. We use 10 principal eigenvectors per class for the MQDF classifier. The parameters of NPC, prototype vectors, are trained using a one-vs-all cross-entropy (CE) criterion. This training objective favors character retrieval as well as text line alignment. We use an NPC with one prototype per class.

### 6.3.1. Effects of geometric models and confidence transformation

Table 4 shows the text line alignment results of our proposed method on all test data (ATData) and the reduced test data without under-segmentation errors (RTData). From the results, we observe that:

1. The geometric models (GMs) can significantly improve the alignment performance on different character classifiers.
2. The confidence transformation (CT) can effectively benefit the combination of geometric models with character classifier.
3. The two character classifiers, MQDF and NPC, perform comparably well in text line alignment, though the NPC has much lower complexity than the MQDF.
4. Removing the text lines with under-segmentation, the proposed method achieves very high accuracy of alignment on RTData. The highest accuracies on two databases CASIA-HWDB and HIT-MW are 99.04% and 98.01%, respectively.

### 6.3.2. Comparing geometric models

To investigate the effects of geometric contexts, we selected 2000 text lines from the test data of CASIA-HWDB. These selected text lines have no under-segmentation errors, but are difficult to be aligned using character recognizer only.

Table 5 shows the effects of geometric contexts \( f_0 = \text{MQDF} \) or \( f_0 = \text{NPC} \). We can see that the incorporation of geometric contexts improve the alignment accuracies remarkably when using any one geometric model or the combination of them. The binary geometric models \( (f_1, f_3) \) perform better than the unary geometric models \( (f_1, f_3) \). This justifies the importance of between-character relationship. Comparing the class-dependent models \( (f_1, f_3) \) and the class-independent models \( (f_2, f_4) \), the results show that the class-independent geometric models perform better. The best alignment performance by combining four geometric models justifies the complementariness of class-dependent and class-independent geometric models. Moreover, the fact that geometric models exhibit almost consistent results on different character classifiers also demonstrates the universality of our geometric models.

### 6.3.3. Comparing with word matching-based methods

There had been few efforts devoted to Chinese handwritten text line alignment before our work, particularly, there had not been character recognition-based methods for this purpose. To justify the superiority of recognition-based alignment, we identified two word matching-based methods that are applicable to Chinese handwritten documents. Obviously, the geometric transformation-based method is not suitable because it was designed for printed documents and can only tolerate rigid deformations. Therefore, in this section, we implemented two word matching methods based on [15,16]. The method of [15] investigates the relationship of word lengths between image and text, and is analogous to the unary geometric models in our work. The method of [16] calculates the distances between connected components, which is similar to the binary geometric models in our work. We accustomed the two methods to suit Chinese text line alignment based on character matching instead of word matching. For the method based on [15], we assume that the punctuation marks, English letters, digits and Chinese characters are of width of 1/10, 1/2, 1/2 and 1 standard character, respectively, and use DTW for exhaustive search to find the best alignment. For the method based on [16], we compute the distance between two overlapped components as in [50], and use greedy search to find the best alignment.

The alignment accuracies of these two methods and our proposed method (using the NPC character classifier) are shown in Table 6. We can see that the proposed character recognition-based method
Table 6
Comparison of the proposed method and word matching-based methods.

<table>
<thead>
<tr>
<th></th>
<th>CASIA-HWDB /ATData/RTData</th>
<th>HIT-MW /ATData/RTData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed method</td>
<td>92.31/99.03</td>
<td>92.66/98.00</td>
</tr>
<tr>
<td>Method based on [12]</td>
<td>63.31/72.89</td>
<td>66.55/71.18</td>
</tr>
<tr>
<td>Method based on [13]</td>
<td>43.19/46.12</td>
<td>46.07/48.41</td>
</tr>
</tbody>
</table>

Fig. 9. Two types of miss errors. (a) A Chinese character is merged into the preceding character. (b) A period is merged into the preceding Chinese character.

Fig. 10. Two types of insertion errors. (a) A dot (part of a Chinese character) is deleted (aligned with non-character “#”). (b) Two mis-written characters are aligned with two non-character “#”; this is a correct deletion though the two mis-written characters were labeled as one non-character in the ground-truth.

Fig. 11. Three types of alignment errors.

6.3.4. Error analysis

Due to the imperfection of pre-segmentation (over-segmentation) and imprecision of character recognition and geometric models, some errors of character segmentation and labeling may remain and all these remaining errors should be corrected manually. The errors can be categorized into three types: miss error, alignment error, and insertion error. A miss error refers to the case that a character in the transcript has no corresponding image segments, due to missed writing or mis-merging the segment of the character with other characters by under-segmentation. Fig. 9 shows two examples of miss errors.

An insertion error refers to the case that a primitive segment has no corresponding transcript character, i.e., it is aligned with a non-character (denoted as “#”). This implies an extra image segment is inserted into the transcript text. Fig. 10 shows two examples of insertion errors. The case in Fig. 10(b) is actually a correct deletion because the deleted characters are mis-written and redundant.

The dominant error, alignment error, includes mis-split of a character into multiple ones and mis-merge of multiple characters into one. In our experiments, we observed three types of alignment errors: (1) Segmentation error between characters (Fig. 11(a)). (2) Heavy overlap of bounding boxes (Fig. 11(b)). (3) Touching strokes between characters are failed to split in pre-segmentation (Fig. 11(c)). To improve alignment for such cases, we need to improve the over-segmentation method.

7. Conclusion

We proposed a recognition-based ground-truthing approach for annotating Chinese handwritten document images. In this approach, we model the single character and between-character geometric contexts and combine with a character recognizer to evaluate the score of candidate segmentations (alignments) of text line image matched with the text transcript under the Bayesian
framework. The optimal alignment is found by dynamic time warping (DTW). Our experimental results demonstrated the superiority of character recognition-based alignment and the benefits of geometric models. Despite that some alignment errors remain, the tool based on the proposed approach can be used for annotating practical handwritten documents with remaining errors corrected by human operators. The large database CASIA-HWDB was annotated using the preliminary version of this tool, and the ground-truth data has been used for design and evaluation of our works in text line segmentation and character string recognition. Further improvements can be expected using better character recognizer and over-segmentation method.

Conflict of interest statement

The authors declare no conflicts of interest.

Acknowledgment

The authors would like to thank Tonghua Su for authorizing us to use the HIT-MW database. This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 61175021 and 60933010.

References


Downloadable at http://www.nlpri.ac.cn/databases/handwriting/CTCLC.html
Qiu-Feng Wang received the B.S. degree in Computer Science from Nanjing University of Science and Technology, Nanjing, China, the Ph.D. degree in Pattern Recognition and Intelligent Systems at the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2006 and 2012, respectively. He is an Assistant Professor at the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include handwriting recognition, sequence pattern recognition, and language modeling.

Cheng-Lin Liu is a Professor at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences, Beijing, China, and is now the deputy director of the laboratory. He received the B.S. degree in electronic engineering from Wuhan University, Wuhan, China, the M.E. degree in electronic engineering from Beijing Polytechnic University, Beijing, China, the Ph.D. degree in pattern recognition and intelligent control from the Chinese Academy of Sciences, Beijing, China, in 1989, 1992 and 1995, respectively. He was a Postdoctoral Fellow at Korea Advanced Institute of Science and Technology (KAIST) and later at Tokyo University of Agriculture and Technology from March 1996 to March 1999. From 1999 to 2004, he was a Research Staff Member and later a Senior Researcher at the Central Research Laboratory, Hitachi, Ltd., Tokyo, Japan. His research interests include pattern recognition, image processing, machine learning, and especially the applications to character recognition and document analysis. He has published over 140 technical papers at prestigious international journals and conferences.