Query Pixel Guided Stroke Extraction with Model-Based Matching for Offline Handwritten Chinese Characters

Tie-Qiang Wang[a,b], Xiaoyi Jiang[c], Cheng-Lin Liu[a,b,d,*]

[a] National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Sciences, Beijing 100190, P.R. China
[b] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, P.R. China
[c] Department of Computer Science, University of Münster, Münster D-48149, Germany
[d] CAS Center for Excellence of Brain Science and Intelligence Technology, Beijing 100190, P.R. China

ARTICLE INFO

Article history:
Received 3 June 2020
Revised 12 August 2021
Accepted 1 November 2021
Available online 6 November 2021

Keywords:
Stroke extraction
Conditional fully convolutional network
PathNet
Stroke matching
Tree search

ABSTRACT

Stroke extraction and matching are critical for structural interpretation based applications of handwritten Chinese characters, such as Chinese character education and calligraphy analysis. Stroke extraction from offline handwritten Chinese characters is difficult because of the missing of temporal information, the multi-stroke structures and the distortion of handwritten shapes. In this paper, we propose a comprehensive scheme for solving the stroke extraction problem for handwritten Chinese characters. The method consists of three main steps: (1) fully convolutional network (FCN) based skeletonization; (2) query pixel guided stroke extraction; (3) model-based stroke matching. Specifically, based on a recently proposed architecture of FCN, both the stroke skeletons and cross regions are firstly extracted from the character image by the proposed SkeNet and CrossNet, respectively. Stroke extraction is solved by simulating the human perception that once given a certain pixel from non-cross region of a stroke, the whole stroke containing the pixel can be traced. To realize this idea, we formulate stroke extraction as a problem of pairing and connecting skeleton-wise stroke segments which are adjacent to the same cross region, where the pairing consistency between stroke segments is measured using a PathNet [1]. To reduce the ambiguity of stroke extraction, the extracted candidate strokes are matched with a character model consisting of standard strokes by tree search to identify the correct strokes. For verifying the effectiveness of the proposed method, we train and test our models on character images with stroke segmentation annotations generated from the online handwriting datasets CASIA-OLHWDB and ICDAR13-Online, as well as a dataset of **R**egularly-**W**ritten online handwritten characters (RW-OLHWDB). The experimental results demonstrate the effectiveness of the proposed method and provide several benchmarks. Particularly, the precisions of stroke extraction for ICDAR13-Online and RW-OLHWDB are 89.0% and 94.9%, respectively.

1. Introduction

Stroke extraction with matching enables offline handwritten Chinese characters to be matched with the structural templates consisting of standard ordered strokes. Stroke extraction is important for structural analysis of handwriting and calligraphy, and applications such as font design and character education. Particularly in educational activities [2], once obtaining strokes with matching correspondence with templates, like the right part of Fig. 1, we can further find irregularly-written strokes (the first case in Fig. 1), missed strokes forgotten by writers (the second case in Fig. 1) and redundantly written strokes (the third case in Fig. 1). In Fig. 1, the reference models of ‘, ‘nd ‘‘ provide the correct shape for each stroke [1,3], but writers may make apparent mistakes because of their irregular writings in early stage of learning. In elementary schools, the checking of irregular writing for students’ written papers poses heavy and tedious burden to teachers. Though Chinese characters are constructed by a set of primitive strokes like the ones in ‘‘ (consisting of dot, horizontal line, turning, vertical line, hook, tick slash, slash and back slash) [4], automatic stroke extraction with matching has not been solved for a long time. Recently, deep neural networks (specially convolutional networks) have yielded very high accuracies (say, 97-98%) on handwritten Chinese character recognition [5–7], which also encourages us to investigate the extraction and analysis of strokes.

Exploiting structural features (strokes and radicals [8,9]) is beneficial for learning and understanding character patterns, but has encounted difficulties. The difficulties of stroke extraction originate

* Corresponding author.
E-mail addresses: tieqiang.wang@nlpr.ia.ac.cn (T.-Q. Wang), xjiang@uni-muenster.de (X. Jiang), liucl@nlpr.ia.ac.cn (C.-L. Liu).

https://doi.org/10.1016/j.patcog.2021.108416
0031-3203/

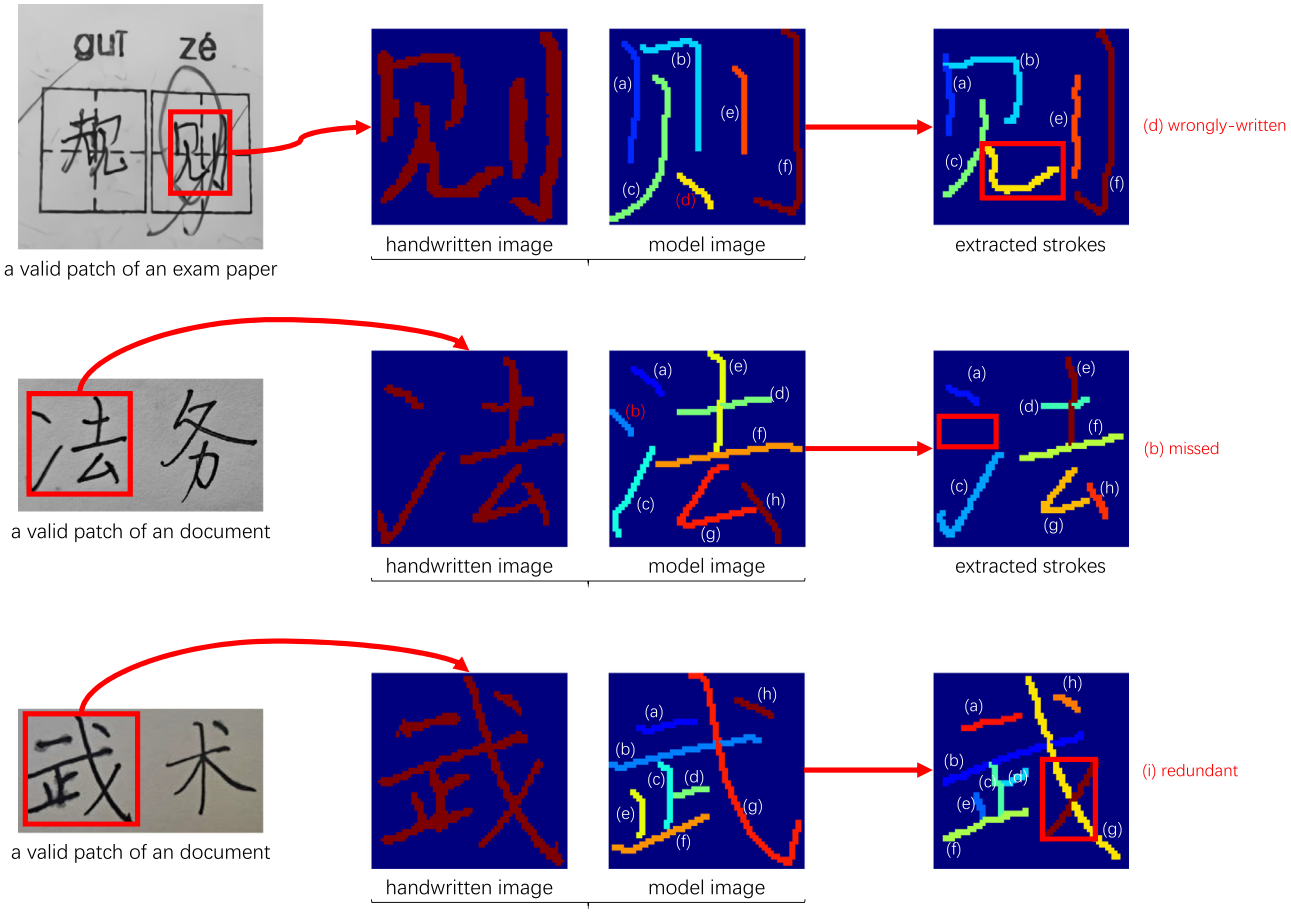

Fig. 1. Some offline handwritten Chinese characters and their reference models. The right part shows the strokes extracted by our method.

from the irregularity and variability of handwriting, the complex stroke shapes (many multi-segment strokes), multi-stroke structure and cross between strokes in Chinese characters. Many efforts have been made for stroke extraction aiming to overcome these problems. The existing stroke extraction methods can be roughly classified into two categories: skeleton-free algorithms [1,10,11] and skeleton-based ones [12–16]. Skeleton-free methods mainly handle machine-printed characters with smooth stroke contours and simple spatial structures. For example, Tseng and Chuang [10] applied stroke-aware rules observed from the structure of machine-printed Chinese characters to extract strokes heuristically. Recently, Kim et al. [1] designed a fully convolutional network (FCN), named PathNet, to calculate the stroke-level consistency between two arbitrary foreground pixels and to extract strokes.

When processing offline handwritten characters with variable stroke widths and cursive writings, analysis on skeletons is more reliable [16–18]. On skeletons, the intersection relationship between strokes can be detected at cross/fork points, and stroke segments can be traced. Based on stroke segments and cross points, character structures can be represented as graphs. Some efforts have been made to deal with the ambiguity of stroke segment connection at cross regions: Lin and Tang [15] detected stroke segments bounded by fork points and connected collinear segments, while Tan et al. [16] derived optimal paths from the degree information.

Though the above methods are equipped with powerful visual rules, extracting strokes from 2D images is still a challenge for most languages. Due to the stroke interference and the ambiguity of curvature, to split and merge line segments into collinear line strokes is not clear-cut [14], especially in local regions which contain multiple candidate stroke segments. Model-based stroke extraction can help overcome this ambiguity. Hsieh and Lee [19,20] used online character models (composed of natural strokes) to guide stroke extraction, where all possible input strokes of the stroke types present in the reference model were detected from the input character, and then the stroke correspondence was obtained by rule-based search; In [21], the line segments of the input and reference characters were initially matched one-to-one and then tentatively connected to form long strokes according to the matching scores during relaxation labeling; differently, Rocha and Pavlidis [22] grouped the stroke components of an input character dynamically to match against a prototype character. Afterwards, Liu et al. [14] considered more character classes and combined stroke-level graph matching with heuristic tree search based stroke extraction. The pre-defined templates in [14] were described in attributed relational graphs with keypoints and line segments, using straight line to approximate all strokes. These methods work well for only neatly written characters, and need large efforts of human design of templates and attributes.

A key issue in stroke extraction is to clarify the stroke-level ambiguity at cross regions, where multiple strokes intersect each other, and among the stroke segments adjacent to a cross region, it is hard to decide which pair of segments form a natural stroke. To overcome this problem, we propose a comprehensive scheme for stroke extraction in offline handwritten Chinese characters. The method consists of three main steps: fully convolutional network (FCN) based skeletonization, query pixel guided stroke extraction, and model-based stroke matching. The stroke extraction part of our method falls in the hybrid class of skeleton-free/skeleton-based methods. Specifically, we first extract stroke skeletons and cross re-

gions by the proposed SkeNet and CrossNet, respectively. Then for extracting complete strokes by connecting stroke segments adjacent at cross regions, we measure stroke consistency by a query pixel guided version of PathNet [1] in a skeleton-free fashion, while the query pixels can be randomly chosen from the skeleton map.

The above steps can extract nearly all strokes, including multi-segment and cross strokes, but inevitably remain ambiguity in connecting adjacent stroke segments. This ambiguity is solved by matching the input character with its reference model consisting of ordered strokes. We collect the stroke templates for thousands of Chinese characters in **Standard Kaiti Font** [23], and use model-based tree search to get the correspondence between handwritten strokes and template strokes while abandoning redundant extracted candidate strokes.

For training the neural networks (SkeNet, CrossNet, PathNet) in this paper, character images and stroke image sequences are produced from online handwritten samples. Experimental results demonstrate that our system provides a feasible solution to extracting and ordering strokes for offline handwritten Chinese characters. On our datasets of unconstrained handwriting and regularly-written characters, the proposed method achieves stroke extraction precision 89.0% and 94.9%, respectively.

In the proposed method, the components SkeNet and CrossNet (which share the backbone but have different output branches) have been reported in our previous conference paper [2,17]. The main contribution of this paper is to present the comprehensive scheme of handwritten stroke extraction, as well as the components of query pixel guided stroke extraction by PathNet and model-based stroke matching. The SkeNet and CrossNet are not described in details, but for completeness of performance evaluation, their experimental results are included in this paper. The proposed method reports superior performance in stroke extraction from offline handwritten Chinese characters. This makes the method applicable in real-world applications such as writing evaluation in education and robot interaction. To our knowledge, the proposed method is the first so far to enable extracting distorted complete (single-segment and multi-segment) strokes from offline handwritten Chinese characters. The proposed method is also applicable to other Asian scripts (Japanese and Korean) which are typical of multi-stroke structure and multi-segment strokes. As for handwritten characters or words of Latin scripts, the proposed SkeNet, CrossNet and PathNet in this paper can be applied to assist stroke trajectory recovery (such as [24]).

The rest of this paper is organized as follows: Section 2 reviews related works; Section 3 presents the overview of method and the preparation of data; Section 4 describes the proposed stroke extraction method; Section 5 presents our experimental results, and Section 6 draws concluding remarks.

2. Related Work

2.1. Skeletonization of Handwritten Characters

Skeletonization, or called as thinning, is an important preprocessing step for character images to facilitate stroke extraction. In the past few decades, two types of thinning algorithms have been proposed: neighborhood-based algorithms [25–27] and distance-based ones [28,29]. Neighborhood-based methods iteratively delete pixels on the stroke boundary until centered lines remain. In contrast, distance based methods yield skeletons in a straightforward manner by using distance transforms to extract the medial axis of strokes. These methods are likely to yield unsatisfactory results when facing: (1) complex shapes, (2) variable stroke widths and (3) unsmooth edges. Particularly, the extracted lines are often distorted at the crosses or intersections of strokes [30]. Recently, FCN based skeletonization has been proven to outperform the above methods remarkably [31], but for training FCN, it is infeasible to label skeleton pixels for millions of offline handwritten samples [32]. Therefore, our previous work proposed to generate annotated skeleton data from online handwritten samples [17].

2.2. Template-Free Stroke Extraction for Chinese Characters

Most stroke extraction methods [1,10,11,33–35] for Chinese Characters firstly analyze ambiguous zones [33] (i.e, cross regions in our paper). For characters in printed fonts, Sun et al. [35] represented them using triangular meshes with their singular regions, then used tangent directions of sub-strokes to highlight ambiguous zones; He and Yan [34] used gradient directions of character contours with chain-code features to detect cross regions. In handwritten Chinese characters, stroke extraction is mostly performed with skeletonization to simplify the analysis of cross regions. For example, the method in [33] even summarized out all the appearances of cross regions in skeleton maps instead of using various rules in [13]. Learning-based stroke extraction methods [1,36] were proposed to cope with complicated cases of cross regions. The method in [36] built a font skeleton manifold and mapped printed characters into their stroke-level labeled variation directly. The PathNet in [1] can extract all strokes in printed characters of Kaiti font [37]. In this paper, we measure the stroke-level consistency using the PathNet, which alone is not sufficient to extract complete strokes from offline handwritten Chinese characters, however.

2.3. Structural Matching with Template for Handwritten Chinese Characters

Structural matching obtains the correspondence between the strokes in input image and the reference strokes in template character. This can largely overcome the ambiguity of stroke extraction. When (candidate) strokes have been extracted from the input character image, the correspondence between input strokes and reference strokes can be formulated as a graph matching problem, which can be solved using relaxation labeling [21,38,39], tree search [14,19,20], or other graph matching algorithms. Reference stroke information was also used to guide the stroke extraction process [20] or to merge stroke segments [14,20–22]. The method in [14] generates multiple candidate strokes from input image by merging stroke segments to match with reference strokes, then searches for the optimal combination of candidate strokes by tree search. This method assumes strokes as straight lines, and so, the extraction of curved, multi-segment and highly distorted strokes remains a big issue. In this paper, we propose techniques for extracting distorted complete (single-segment and multi-segment) strokes and use structural matching to solve the ambiguity in stroke extraction.

3. System Overview and Data Representations

3.1. System Overview

Our stroke extraction system, shown in Fig. 2, is built combining the following techniques: (1) extracting stroke segments from skeleton generated by the SkeNet and cross regions using the CrossNet; (2) using the PathNet [1] to model the stroke-level consistency between stroke segments adjacent to the same cross region; (3) character model based stroke matching [14] to disambiguate how stroke segments are connected.

Here, SkeNet, CrossNet and PathNet are all trainable FCNs, and provide stroke segments for the stroke matching step. The SkeNet and the CrossNet share the same backbone part [6] and extract the skeletons/cross regions, respectively. The PathNet has two input channels: the original input image and the query pixel map,

and is trained to find the stroke containing the query pixel provided by the other input channel. For a pair of stroke segments, the bi-directional query result between two query pixels (one on each stroke segment) can model the consistency between two segments [1].

In training the FCN models, the SkeNet and the CrossNet, sharing a backbone, are trained jointly. The PathNet, with two-channel input, is trained separately. The training data, with ground-truths of skeleton/cross pixels (for SkeNet/CrossNet) or complete strokes (for PathNet), are generated automatically from online handwritten characters. The details are described in Section 3.2.

Based on the pairing consistency measured by the PathNet, the strokes extracted by connecting stroke segments adjacent to the same cross region still remain ambiguous, especially when a stroke segment can be connected with multiple adjacent segments or a stroke consists of more than two segments. This ambiguity is solved by model-based stroke matching: the extracted candidate strokes from input image are matched with a character model consisting of ordered standard strokes by tree search. The character models are generated from the public Make Me A Hanzi dataset [37]. If the character model is of the same character class with the input handwritten character, then most strokes in the input character can be matched with the standard strokes in the model. The class of the input character can be reasonably assumed known based on two assumptions: (1) in some application scenarios, the users are asked to write characters of specified classes; (2) currently, handwritten characters (even cursive writing and slightly wrong writing) can be recognized with high accuracies by deep neural networks [5].

3.2. Data Preparation

The supervised training of our models (SkeNet, CrossNet and PathNet) demands plenty of ground-truthed skeletons, cross regions caused by intersecting strokes, and separated strokes of offline handwritten Chinese characters. However, it is almost impossible to obtain millions of samples with pixel-wise manual annotations.

Fortunately, online handwritten characters record strokes by (x, y)-coordinate sequences [32], which can be viewed as the ideal skeletons of synthesized images (generated by dilating the stroke skeletons) [17]. Thus, we generate a large number of synthesized character images from online handwritten characters to train all the models in this paper. The generated offline images inherit the character shapes and writing style of online characters, so the negligible visual differences between the synthesized data and the real offline data ensure the effectiveness of our models [40].

As shown in Fig. 3, each online sample records its strokes in writing orders, thus, all stroke images can be generated by dilating the strokes of a plotted online character image with appropriate control of stroke width (randomly sampled) and edge smoothness [17]. This step outputs offline stroke images $s_1, ..., s_n$, which are generated in writing order and listed in the second row of Fig. 3. Next, the whole character image is the union of $s_1, ..., s_n$, and the ground-truthed cross regions are formed by the pixels which are shared among multiple strokes.

3.2.1. Pre-Processing of Online Handwritten Data

An online handwritten Chinese character sample of class $\mathcal{C}$ is recorded as the stroke trajectory set $\mathbf{S} = \{\mathrm{S}^1, ..., \mathrm{S}^n\}$. We use anisotropy resizing to normalize all the coordinates in $\mathbf{S}$ into a 64×64 area, and obtain $\mathbb{S} = \{\mathcal{S}^1, ..., \mathcal{S}^n\}$, where $\mathcal{S}^i = \{(x_1^i, y_1^i), ..., (x_{n_i}^i, y_{n_i}^i), <\texttt{eos}>\}$. To remove the redundant points in the original data, we propose a simple strategy to pre-process the online stroke sequences like that in [41]. Specifically, in $\mathcal{S}^i$, whether to remove a point (x, y) or not depends on two conditions: the distance D of this point away from its former point, the angle A constructed by this point and its two neighbors. We decide that (x, y) is a redundant point and remove it once $D < 1.5$ or ($D < 5$ and $A > 150^\circ$) (considering that removing the point causes little deviation of stroke shape).

Note that the size normalization (character re-scaling to 64×64 in this paper) operation makes character recognition or stroke extraction scale-invariant. On the other hand, rotation invariance is not encouraged in character recognition, because rotated characters of different classes may be confused (such as digits 6 and 9). Nevertheless, tolerance to small rotation is necessary, because handwritten characters normally have rotation or skewness. This is offered by the tolerance of shape matching or stroke matching measures.

Besides, in the CASIA-OLHWDB [32] and ICDAR13-Online [42] datasets, a large number of handwritten strokes are joined-up, which means one saved trajectory contains multiple strokes. These joined-up strokes are brought out because of the undue connections between adjacent strokes in cursive writing. Such connected strokes will cause errors in generated ground-truthed images, in that multiple strokes are labeled as one stroke. To solve this problem, we try to detect the connected strokes according to the pattern of turning between stroke segments. Generally, in the Cartesian coordinate system, the directions determined by two adjacent line segments $\mathcal{S}_{[k-1]}^i \to \mathcal{S}_{[k]}^i$ and $\mathcal{S}_{[k]}^i \to \mathcal{S}_{[k+1]}^i$ probably locate in one of four quadrants, and the trajectory of $\mathcal{S}_{[k-1]}^i \to \mathcal{S}_{[k]}^i \to \mathcal{S}_{[k+1]}^i$ varies in 16 different cases, among which five cases go beyond the standard writing habits of Chinese characters. For such abnormal trajectories as shown in Fig. 4, the stroke trajectory should be split into multiple strokes at $\mathcal{S}_{[k]}^i$. Otherwise, like Fig. 5(b), the lower-right trajectory containing two connected strokes causes an incorrect ground-truthed image. To obtain more accurate annotations, we split the joined-up trajectories meeting with the above conditions in the CASIA-OLHWDB and ICDAR13-Online. Note that we only split strokes at obviously abnormal turning points (which are highly probable to join up different strokes), while the normal turning parts (such as the top-right corner of Fig. 5(c), which is a regular multi-segment stroke) are remaining un-split.

3.2.2. Data Generation for SkeNet and CrossNet

By linking all the stroke segments onto a 64×64 plane ((a) step in Fig. 3), skeleton-wise stroke images I_s^t can be readily produced and serve as the training target of the SkeNet. As the (b)/(c) steps in Fig. 3 show, when drawing stroke image sequence $\{s_1, ..., s_{n_i}\}$, we assign each stroke in $\mathbf{S}$ a random width sampled from the Gaussian distribution $\mathbf{N}_{\mathcal{C}}$, then resize the generated images into 64×64-size. The mean value and variance of $\mathbf{N}_{\mathcal{C}}$ are induced from the widths of real offline characters of class $\mathcal{C}$ in the offline dataset CASIA-HWDB datset [32]. The method of calculating stroke width is the same as [14]: for each character image, calculates the area of foreground pixels, takes the half contour length as stroke length, then the ratio of area to length is the estimated stroke width.

Besides, we define all the overlapping pixels owned by more than one stroke as cross pixels. The training target of CrossNet is to predict the overlapping pixels in a character image.

3.2.3. Data Generation for PathNet

The forward computations of the PathNet require ground-truthed data shown as the (e) step in Fig. 3, where we keep a **non-cross** foreground pixel in the offline character image and set all the rest pixels as zero to produce the input query map (like the dark map with a red pixel in Fig. 3), and the other one input channel for the PathNet is filled by the whole character image). Hence, the training target of this query pixel is

Table 1
Notations.

Notation	Description	Notation	Description
I	original input image	f_p	function of PathNet
f_c	function of CrossNet	$p_q \in S_T$	query pixel (red in Fig. 2)
I_c	cross map $= f_c(I)$	$p_u \in S_T$	undetermined pixel when knowing p_q
I_c^t	target of CrossNet	M_c	set of stoke segments for stroke matching
f_s	function of SkeNet	g_i	i-th stroke segment in M_c
I_s	skeleton map $= f_s(I)$	S_T	final result of skeletonization
I_s^t	target of SkeNet	$M_R = \{r_1, ..., r_{N_k}\}$	reference model with N_R strokes

to predict the stroke image which contains this pixel. In summary, for a character sample with N strokes, we randomly choose non-cross query pixels for each stroke and provide N {input query map, character image, target stroke image} triplets to serve as the training data of the PathNet.

3.2.4. Building Reference Models for Stroke Matching

Fig. 2 shows the role of reference character models. The reference models can be obtained by parsing automatically from a free and open-source dataset named **Make Me A Hanzi** [37,43], in which more than 9,000 Chinese characters were collected by ARPHIC Company [3] with their standard skeleton-wise strokes. Our data parser is illustrated in Fig. 6, where the original data is saved in (1024,1024)-size coordinates as the blue box shows.

4. The Proposed Stroke Extraction Method

The description of the proposed method involves many notation symbols, which are summarized in Table 1. As shown in Fig. 1, the proposed method has three main steps: skeletonization using the SkeNet, query pixel guided stroke extraction, and model-based stroke matching. The details of the skeletonization have been reported in [17]. This paper focuses on the latter two steps. The CrossNet will be introduced when describing query pixel selection.

4.1. Query Pixel Guided Stroke Extraction

After skeletonization of character image, the stroke segments delimited by end points and crossing points can be traced, and the task of stroke extraction amounts to the joining of stroke segments adjacent to a cross region. We use the PathNet to measure the consistency of joining a pair of adjacent stroke segments.

4.1.1. PathNet for Modeling Stroke-Level Consistency

The objective of modeling stroke-level consistency is to measure how likely two different stroke segments belong to the same stroke. To do this, the PathNet (as shown in Fig. 2) has two input channels: the query pixel map I_q, and the character image I. The image I_q has one pixel (the position of the query pixel p_q) labeled as 1 and all the other pixels as 0. At the position of an undetermined pixel p_u, the PathNet outputs $I_u[p_u] = f_p(I_q, I)[p_u]$ [1].

The image I provides the whole search space for I_q to find its congeneric pixels. To optimize $f_p(I_q, I)$, the PathNet is trained to minimize the cross entropy loss:

$$L_p = -\sum_i y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i), \quad \hat{y}_i = \sigma(f_p(I_q, I)[p_i]), \quad (1)$$

where σ is the sigmoid function, p_i is the i-th pixel in the output map of PathNet, and y_i is the value at the position of p_i in the target image, which indicates that whether p_i and p_q belong to the same stroke or not (i.e., the pixels locating in the same stroke with p_q are labeled as 1 while all the other pixels are labeled as 0). $\hat{y}_i$ predicts the relationship between p_i and p_q.

In our experiments, the PathNet shares the same architecture with the VDSR Net in [44], which consists of 32 repeated conv+ReLU+BatchNorm blocks, which are all of 64 3×3-size and 1-stride filters.

4.1.2. How to Choose Query Pixels

Obviously, a query pixel p_q should be located in a stroke segment. To ensure that each query pixel can trace to an unambiguous stroke, we avoid selecting query pixels from the cross regions. Based on the output map of the CrossNet, we detect and exclude all pixels at the cross regions of strokes before choosing query pixels. Hence, we ensure that p_q belongs to one and only one stroke. The main rules for choosing query pixels are:

- All representative pixels (query pixel and undetermined one) we need are from the skeleton map S_T. The skeleton-wise strokes in S_T are the central lines of the original strokes. It has been proven and visualized that S_T is most reasonable for describing handwritten character images [2].
- For any two stroke segments that meet at the same cross region o_i, though we exclude the two query pixels p_q and p_u from o_i, we still expect that p_q and p_u are as spatially close as possible. As a conditional FCN, $f_p(I_q, I)$ tends to judge the relationship between p_q and p_u more accurately when p_u is close to p_q.

4.1.3. How to Calculate S_T

As shown in Fig. 7, the cross map I_c and the skeleton map I_s are generated via two isomorphic FCN branches proposed by [17,31], whose configurations are designed in accordance with the building rules proposed by DeepSke Net [31]. The parameters of these two networks are as follows: 1) convolution filters initialized by HCCR-CNN9Layer [6]; 2) weights for working out the side outputs from multiple scales (the two branches in Fig. 7) [45]; 3) learnable bilinear filters [17] for enlarging feature maps; 4) convolution kernels for generating candidate maps (the 4th row in the SkeNet/CrossNet branch shown in Fig. 7) and the convolutional fusion. Here, dilated convolution is used to maintain high resolution of feature maps in FCNs through replacing the max-pooling operation or convolution layer [46], and it can also expand the receptive field. Besides, the 3rd and 4th rows in the SkeNet/CrossNet branch of Fig. 7 illustrate the recombination of feature maps.

When training the SkeNet/CrossNet, we deploy the same binary cross entropy loss of Eq. (1). To obtain better skeleton, we adopt the post-processing technique introduced in our previous paper [17]: using $K-Means$ clustering on the $\sigma(I_s)$ ($K=2$, σ means the sigmoid function) to calculate a threshold, which transforms $\sigma(I_s)$ into a binary image, and the ZhangSuen Algorithm [25] to clear the redundant foreground pixels and generate ideal skeletons. While for calculating the final cross map, the image $\sigma(I_c)$ is simply binarized using threshold 0.5 (median value of $\sigma(I_c)$). $K-Means$ clustering is not used in this case because cross and non-cross pixels are severely imbalanced. For convenience, the intermediate results of skeletonization and the final result of cross detection at this stage are still denoted as I_s and I_c, respectively.

After obtaining I_c and I_s, we represent each cross region o_i of I_c by its centroid point p_i^c, connect all the adjacent skeleton pixels in o_i with p_i^c to produce the final skeleton map S_T. This is to say, S_T is

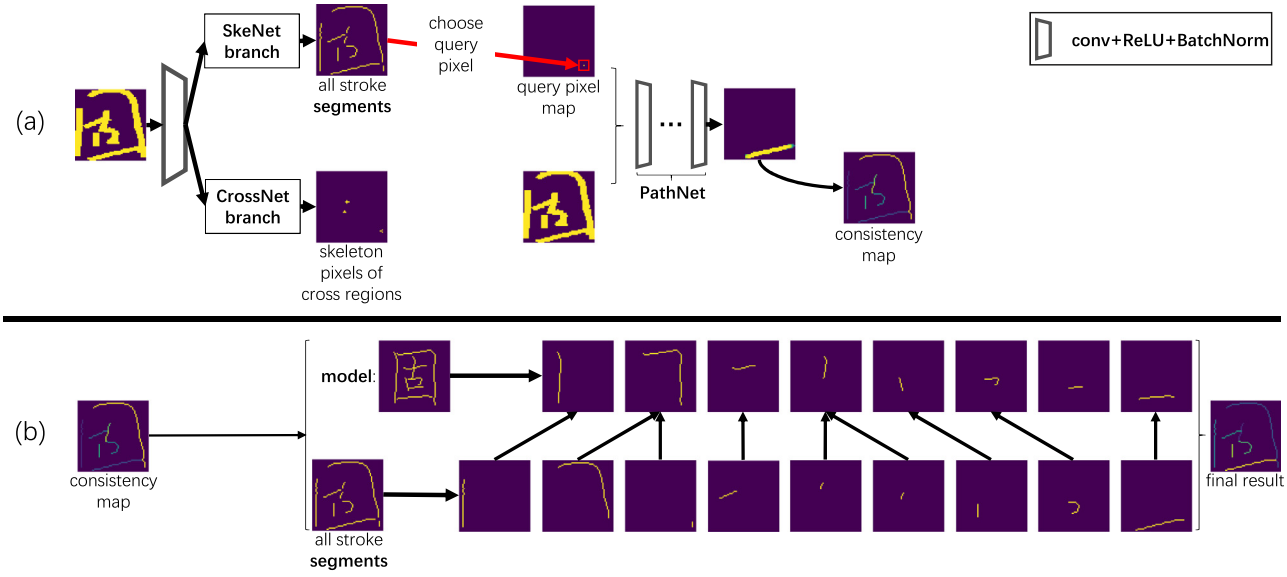

Fig. 2. The pipeline of the proposed stroke extraction system.(a) Skeletonization using SkeNet and CrossNet, and stroke-level consistency modeling using PathNet; (b) Candidate stroke extraction and model-based matching.

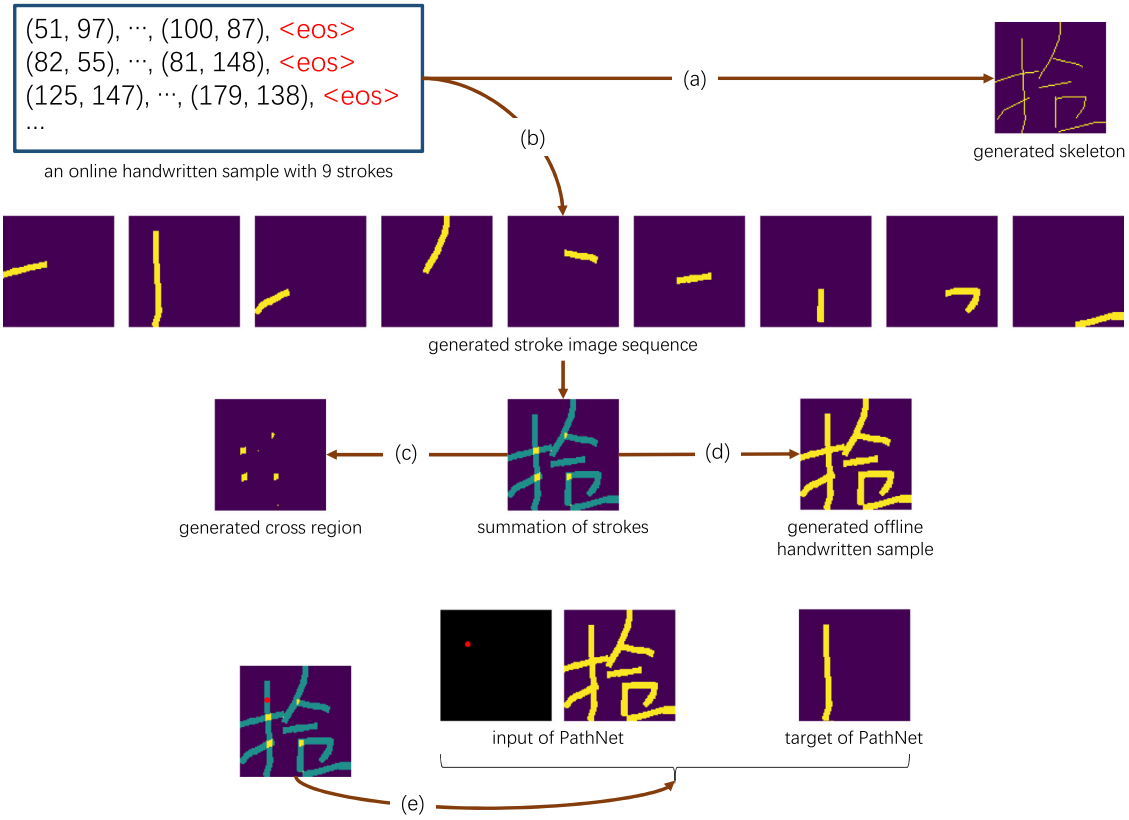

Fig. 3. Synthesizing methods for the training/test data in this paper (<eos> means **e**nd **o**f **s**troke). Brown arrows indicate the steps of automatic generations, and all images share the same size 64×64. The whole pipeline consists of five steps: (a) drawing the skeleton maps; (b) saving all strokes with random widths; (c) generating cross regions; (d) generating offline handwritten samples; (e) data generation for PathNet.

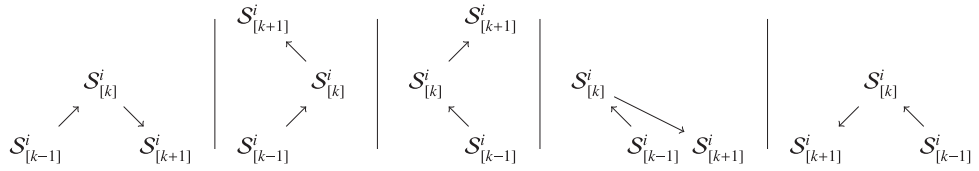

Fig. 4. Five types of abnormal trajectories consisted by $\mathcal{S}_i^{[k-1]}$, $\mathcal{S}_i^{[k]}$ and $\mathcal{S}_i^{[k+1]}$. The arrows point the directions of the handwritten trajectories.

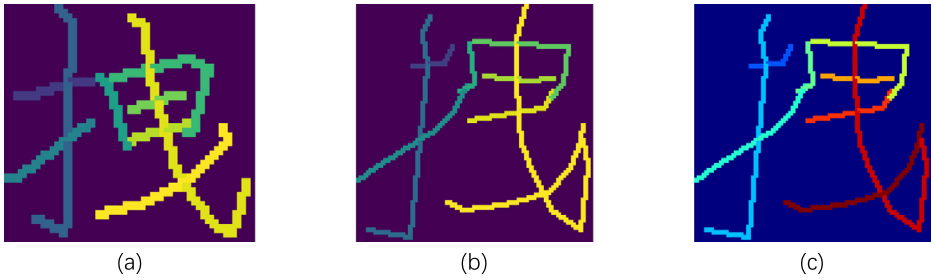

Fig. 5. The necessity of splitting joined-up trajectories. (a) is a standard template character with nine strokes. (b) provides the generated ground-truthed image without splitting. (c) displays the correct ground-truthed image after splitting. Different colors indicate different strokes.

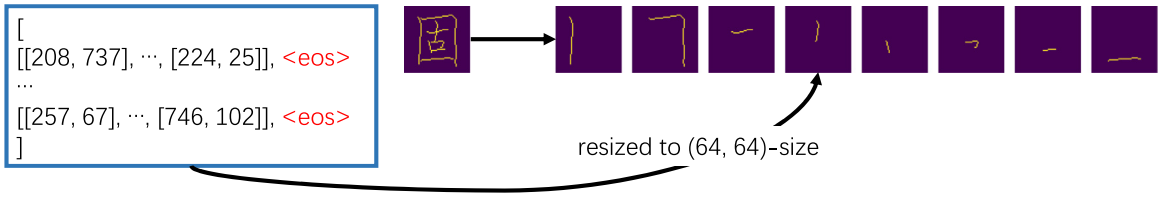

Fig. 6. A sample of pre-defined reference model, consisting of ordered standard strokes.

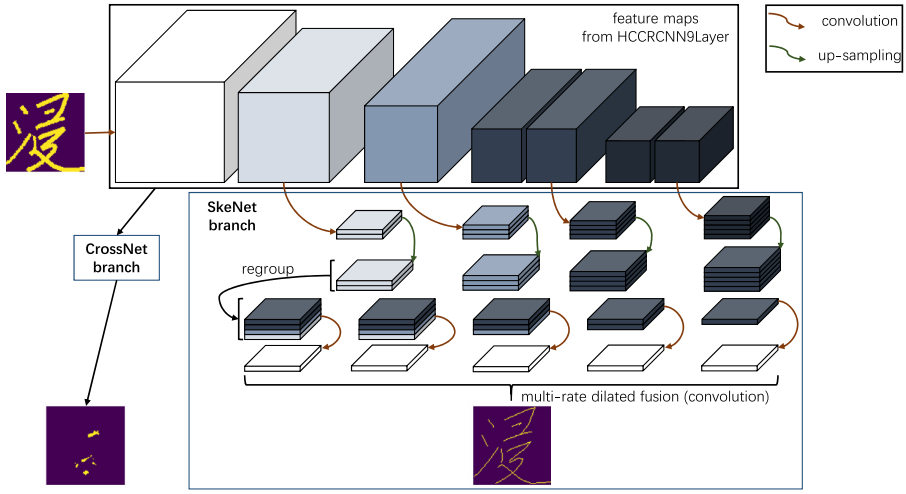

Fig. 7. Data flow of the cross/skeleton map generation using the SkeNet and CrossNet in Fig. 2. Here we omit the 2×2-max-pooling and batch normalization layers after each convolutional group.

produced by re-connecting each skeleton segment to the centroid of its adjacent cross region. This re-connection step is to reduce the distortion of skeleton at cross regions.

4.1.4. The Stroke Extraction Phase

The stroke segments generated by S_T are the basic units in our stroke extraction method. The stroke segments adjacent to the same cross region are judged to whether to merge or not. By connecting the mergeable stroke segments, we obtain complete multi-segment strokes. In cursive handwriting, it often happens that multiple regular strokes are connected by ligature. While the CrossNet is designed to detect cross regions (intersections of strokes), it may not detect the turning points where two strokes are connected. So, to guarantee that different strokes are split at turning points, we use the corner detection technique of [47] to locate the corner points on stroke segments, and split at corner points with angle $< 120^\circ$ (this threshold is to guarantee that most stroke connection corners can be detected). Though this may also split regular multi-segment strokes at turning points, the later process of stroke segment merging will re-join the mis-split stroke segments.

To judge whether to merge two stroke segments adjacent to the same cross region or not, we use the PathNet to measure the consistency score of two adjacent stroke segments. The query pixel guided stroke extraction procedure is illustrated in Algorithm 1 to find the mergeable stroke segments at o_i. First, all the stroke segments adjacent to o_i are traced. Then, each valid pair of segments with a consistency score (higher than a pre-defined threshold) is temporarily reserved. As shown in Algorithm 1, $p_q \rightarrow p_u$ indicates that, under the guidance of p_q, we should analyze that whether p_u belongs to the same stroke as p_q or not. Necessarily, $p_u \rightarrow p_q$ is also considered. Based on these, the consistency score for pairing two stroke segments is defined as

$$s_{q \rightleftharpoons u} = \frac{I_u[p_u] + I_q[p_q]}{2}. \tag{2}$$

As illustrated in Section 4.1.1, both $I_u[p_u]$ and $I_q[p_q]$ are sigmoid output of PathNet. So, the value of $s_{q \rightleftharpoons u}$ is in (0,1). Among the valid pairs (with sufficient consistency score ≥ 0.95) that share common segments at o_i, we merge the stroke segments in the pair with the highest score. Here, $s_{q \rightleftharpoons u} \geq 0.95$ ensures that all the merged segments in our training dataset are correct.

Algorithm 1 Query Pixel Guided Stroke Extraction

INPUT: S_T, I, and $M = \varnothing$, $M_c = \varnothing$
obtain all the centroid points of cross regions $P^o = \{p_1^o, ..., p_{N_o}^o\}$, N_o is the number of detected cross regions
for $p_i^o \in P^o$ **do**
  construct a set $M_i = \varnothing$ of mergeable stroke segments provided by p_i^o
  from S_T, collect the set of stroke segments $G = \{g_1, ..., g_{N_s}\}$ ending at p_i^o
  save $\binom{N_s}{2}$ pairs from G_i in set S_p
  for each pair $[g_q, g_u] \in S_p$ **do**
    select two query pixels p_q and p_u from g_q, g_u, respectively
    given query pixel p_q and I, the PathNet outputs $I_u[p_u]$ at the position of p_u
    given query pixel p_u and I, the PathNet outputs $I_q[p_q]$ at the position of p_q
    calculate consistency score $s_{q \rightleftharpoons u} = \frac{I_u[p_u] + I_q[p_q]}{2}$
    if $s_{q \rightleftharpoons u} \geq 0.95$ **then**
      put the pair (p_q, p_u) into M_i
    end if
  end for
  if $M_i = \varnothing$ **then**
    continue
  end if
  do
    move the pair P_{best} with the highest score from M_i to M
    delete other pairs sharing common stroke segments with P_{best} from M_i
  until $M_i = \varnothing$
end for
pairs with common segments form a new stroke segment
each independent pair (having no common segments with all the other pairs) in M forms a new stroke segment
put all the newly-formed stroke segments and other unpaired ones into M_c
RETURN: M_c

When evaluating our stroke extraction method **without** stroke matching, the output stroke segments of Algorithm 1 are the result of stroke extraction. For stroke extraction **with** stroke matching, the only difference is that 0.5 instead of 0.95 is supposed to be an adequate threshold of $s_{q \rightleftharpoons u}$. This is to let ambiguous pairs of stroke segments to be solved in stroke matching instead of relying on consistency score only.

4.2. Reference Model Based Stroke Matching

In Algorithm 1, the consistency score $s_{q \rightleftharpoons u}$ given by the PathNet tells how likely two adjacent stroke segments should be connected into one segment. Though the PathNet alone can extract most strokes, the lacking of the guidance from character models can still result in ambiguous or wrong stroke at complex cross regions. In this section, we aim to solve the ambiguity of stroke segment connection by stroke matching with character models.

Stroke matching aims to search for the most plausible input strokes corresponding to the ordered reference strokes. Suppose the reference model $M_R = \{r_1, ..., r_{N_R}\}$ has N_R strokes, and an equal number of input strokes $\{s_1, ..., s_{N_R}\}$ are extracted from the input character image I, the distance D between I and M_R is

$$D(M_R, I) = \sum_{i=1}^{N_R} d(r_i, s_i), \tag{3}$$

where d measures the stroke matching distance. Our goal is to extract a combination of strokes from I so that the distance $D(M, I)$ is minimized. From Section 4.1, we already obtain all stroke segments, whose valid subsets can be formed into multiple candidate strokes for each reference stroke r_i. In this section, we follow the standard writing orders of the strokes in the pre-defined character models and use heuristic search [48] to seek for the optimal solution.

Specifically, we formulate stroke matching as a combinatorial optimization problem and solve it using the heuristic A^* search [49]. A^* algorithm accomplishes stroke matching in two stages: candidate stroke extraction and tree search based matching. Based on the extracted candidate input strokes, the search space of stroke matching is represented as a tree structure. We denote the j-th candidate stroke (a merged subset of stroke segments) of reference stroke r_i as node N_{ij} in the search space, and use the matching cost G between N_{ij} and r_i as $d(r_i, s_j)$.

As shown in Fig. 8, the reference strokes are listed from up to down in the original writing order which also determines the searching order. Meanwhile, $CAND_i$ provides all candidate nodes in the search space for a reference stroke r_i. Thus, the j-th candidate node $N_{ij} = \{g_1, .., g_k, ..., g_n\}$ (composed of n stroke segments) of reference stroke r_i in the tree stands for a input-reference stroke pair $\{g_1, .., g_k, ..., g_n\} \leftrightarrow r_i$, and causes a matching cost $G(N_{ij})$ once being searched.

In A^* search, for a partial (i.e., currently searched but incomplete) path P from root and ending at node N_{ij}, we accumulate the matching cost of all the matched (searched) nodes in P as its matching cost $G(P)$. Besides, A^* search also provides a heuristic function $H(N_{ij})$ to estimate the remaining cost of a complete solution containing P. Hence, the overall cost $C(P)$ of P is

$$C(P) = C(BEGIN - \cdots - N_{ij}) = \sum_{N \in P} G(N) + H(N_{ij}), \tag{4}$$

which can be regarded as $D(M_R, I)$ in Eq. (3) and minimized by A^* search. Here, *BEGIN* denotes the root node in search space.

4.2.1. Candidate Stroke Extraction

Once obtaining all the stroke segments $\{g_1, ..., g_j, ..., g_{N_g}\}$ (including the merged stroke segments by Algorithm 1), we need to match an input stroke (formed by a subset of stroke segments) with each reference stroke. Therefore, for each reference stroke, we need all the candidate input strokes which are possibly matched with it.

In Algorithm 2, we use $M_R = \{r_1, ..., r_{N_R}\}$ to denote the reference model with N_R reference strokes, and extract candidate strokes (i.e., subsets of stroke segments) from M_c (output by Algorithm 1) for each reference stroke r_i. $MinDis(r_i, g_j)$ in Algorithm 2 measures the shortest distance between the reference stroke r_i and stroke

Algorithm 2 Candidate Subsets of Stroke Segments (Stroke) Extraction

INPUT: M_c (saving all the stroke segments), M_R
for $r_i \in M_R$ **do**
  $SEG_i = \varnothing$
  for g_j in M_c **do**
    if $MinDis(r_i, g_j) \leq 6$ **then**
      put g_j into SEG_i
    else if SEG_i has stroke segments connected to g_j **then**
      put g_j into SEG_i
    end if
  end for
  $CAND_i = \{\varnothing\}$
  put **all** the feasible subsets of SEG_i into $CAND_i$
end for
RETURN: $CAND = \{CAND_1, ..., CAND_{N_R}\}$

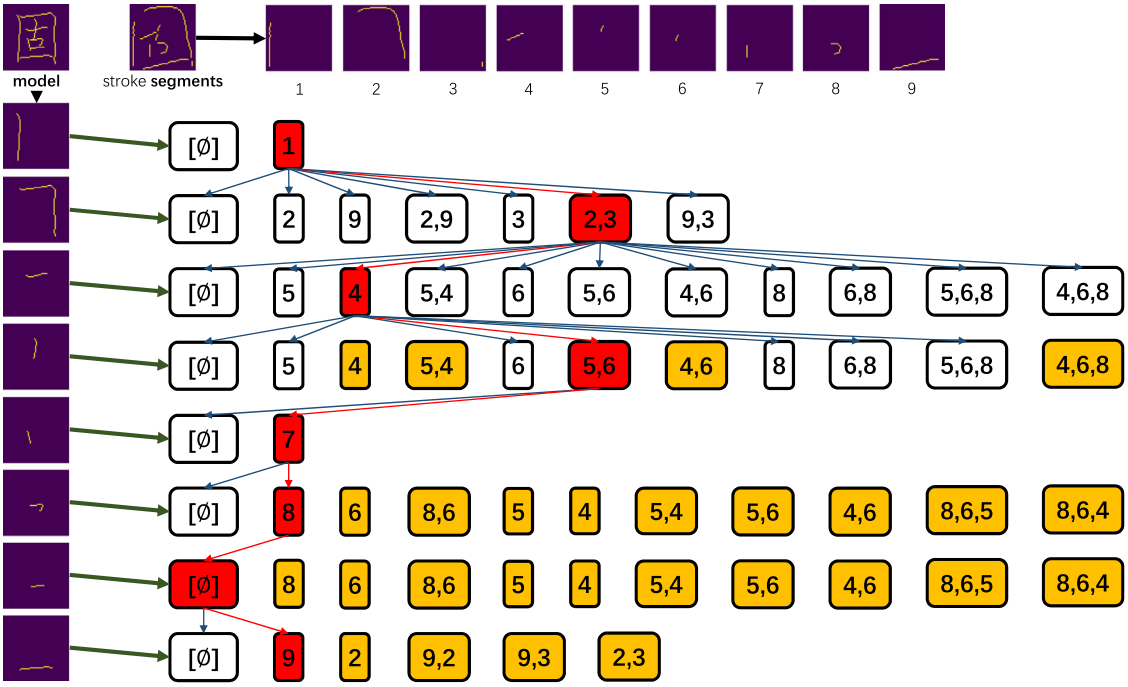

Fig. 8. An example of tree search for stroke matching, where each box denotes a candidate node in the tree. Red boxes with arrows show the nodes in the final searched path, and white boxes show the other candidate nodes with larger cost. Yellow boxes show the incompatible nodes with the searched path. Here, we omit the cost-free root node *BEGIN* and goal node *END*.

segment g_j. For r_i, we first choose all the stroke segments with $MinDis(r_i, g_j) \leq 6$ (compared to normalized character width/height 64), and put them into SEG_i. Then, we iteratively put other segments into SEG_i which are adjacent with at least one of the chosen segments in SEG_i. Finally, from SEG_i, we form all the feasible subsets of segments into $CAND_i$ following two rules: (1) all stroke segments in one subset should build one and only one connected component; (2) merging more than two segments ending at the same cross region is avoided. Besides, $CAND_i$ in Algorithm 2 must contain one empty subset, which indicates that there is no input stroke matched with r_i. The extracted candidate strokes are shown in Fig. 8, where all the boxes are the candidate nodes in the tree structure, and all the candidate nodes for r_i are put in the i-th row.

4.2.2. Tree Search in Model-Based Stroke Matching

In Algorithm 3, we show the whole searching process of our method. To initiate the A^* search, we generate a root node *BEGIN*, whose branches point to all the candidate nodes of r_1. Meanwhile, a goal node *END* is regarded as the stopping sign of searching and pointed by all the candidate nodes of the last reference stroke r_{N_R}. Moreover, both *BEGIN* and *END* are cost-free. *BEGIN* is initially stored in the list *OPEN*, and the list *CLOSED* is initially empty. After selecting the best node from *OPEN* which has the minimum path cost (Eq. 4) with its reference stroke r_{cur}, we expand this node with all the candidate nodes of the next reference stroke.

During searching, one stroke segment belongs to only one reference stroke at most, but some newly-touched nodes (depicted in yellow in Fig. 8) contain repeated stroke segments with the searched nodes in P. In this case, they are treated as incompatible with P and should not be expanded.

Algorithm 3 A^* search for stroke matching

```
INPUT: OPEN, CLOSED
add BEGIN with cost= 0 into the tree and OPEN, BEGIN points to all nodes of r_1
add END with cost= 0 into the tree, and all nodes of r_{N_R} point to END
while OPEN ≠ ∅ do
    choose N_cur with minimal C(BEGIN − ··· − N_cur) from OPEN, N_cur belongs to CAND_cur of r_cur
    if N_cur = END then
        trace from N_cur back to BEGIN, and obtain complete path P
        RETURN P
    else
        move N_cur into CLOSED
        for N in CAND_{cur+1} do
            if N ∉ CLOSED and N ∉ OPEN then
                set parent node of N as N_cur, put N with C(BEGIN − ··· − N_cur − N) into OPEN
                if N ∈ OPEN then
                    if C(BEGIN − ··· − N_cur − N) < C(BEGIN − ··· − N) then
                        change the parent node of N as N_cur
                    end if
                end if
            end if
        end for
    end if
end while
```

4.2.3. Calculating $G(N_{ij})$

In the searched path, N_{ij} matches a candidate stroke with r_i, and causes a matching cost. Therefore, we design a simple function $G(N_{ij}) = G_d(N_{ij}) + G_a(N_{ij}) + G_p(N_{ij})$ to measure this cost. Specifically, G_d tells the spatial distance between N_{ij} and r_i, and G_a measures the directional difference between them. G_p calculates the consistency score based cost of the stroke formed by N_{ij}. For the

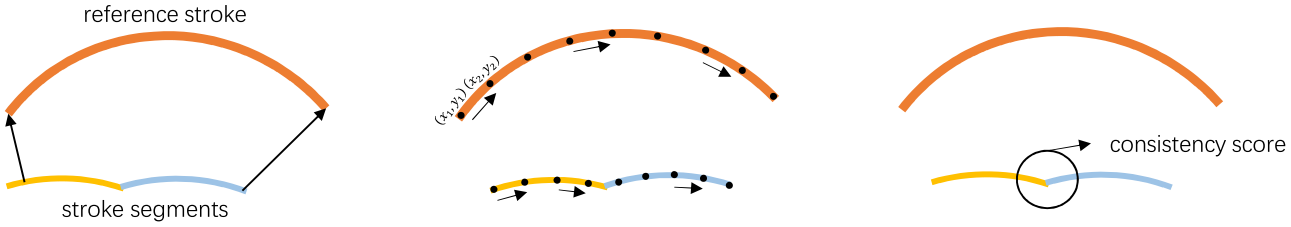

Fig. 9. The calculations of G_d, G_a and G_p.

j-th candidate node $N_{ij} = \{g_1, .., g_k, ..., g_{n^i}\}$ of reference stroke r_i, its cost is

$$\begin{aligned} G(N_{ij}) &= G_d(N_{ij}) + G_a(N_{ij}) + G_p(N_{ij}) \\ &= \frac{1}{n^i}\sum_{k=1}^{n^i} MinDis(g_k, r_i) + \Big[1 - C_s(N_{ij}, r_i)\Big] \\ &\quad + \frac{1}{n_m}\sum_{v=1}^{n_m}\Big[1 - s_{q\rightleftharpoons u}(P_v^i)\Big], \end{aligned} \tag{5}$$

where n^i is the number of stroke segments composing the candidate stroke, and n_m is the number of segment pairs to be merged. $MinDis(g_k, r_i)$ in G_d measures the minimal distance between the stroke segment g_k and the reference stroke r_i, C_s measures the directional similarity, and $s_{q\rightleftharpoons u}$ measures the merging consistency of a pair of stroke segments (if the pair of stroke segments join at a turning point other than a cross region, $s_{q\rightleftharpoons u}$ takes 1 for cost-free merging). For balancing the scale of different costs, $MinDis(g_k, r_i)$ is normalized to [0,1] by dividing $64\sqrt{2}$ (the diagonal length of the input image). P_v^i means the v-th pair of adjacent stroke segments in N_{ij}, which merges n_m pairs of segments sequentially. In the left part of Fig. 9, the black arrows indicate all the shortest lengths from the stroke segments $g_1, .., g_k, ..., g_n$ to the reference stroke r_i. In general, freely-handwritten strokes cannot conform with their corresponding reference strokes perfectly because of their variable positions. Therefore, $MinDis(g_k, r_i)$ reduces the cost caused by some strokes with correct shape but positional deviations.

As the black points in the middle part of Fig. 9 show, we use cosine similarity $C_s(N_{ij}, r_i)$ to model the directional similarity between N_{ij} and r_i as follows: (1) a point sequences $\Lambda(r_i)$ is uniformly sampled from r_i; (2) represent $\Lambda(r_i) = \{(x_1, y_1), ..., (x_k, y_k), ...\}$ with its direction vector $\Phi(r_i) = [\Delta x_1, \Delta y_1, ..., \Delta x_k, \Delta y_k, ...]$, where $\Delta x_k = \frac{x_{k+1}-x_k}{d_k}$, $\Delta y_k = \frac{y_{k+1}-y_k}{d_k}$, and $d_k = \sqrt{(x_{k+1}-x_k)^2 + (y_{k+1}-y_k)^2}$. Similarly, we also obtain $\Phi(N_{ij})$ as the direction vector of N_{ij}. And finally, the similarity is calculated as

$$C_s(N_{ij}, r_i) = \frac{1}{2}\frac{\Phi(N_{ij}) \cdot \Phi(r_i)}{||\Phi(N_{ij})|| \times ||\Phi(r_i)||} + \frac{1}{2}, \tag{6}$$

Obviously, larger directional similarity should be accompanied with smaller matching cost. Therefore, $G_a(N_{ij})$ is determined by $1 - C_s(N_{ij}, r_i)$. In our system, $G_a(N_{ij})$ can not only represent the global trends of strokes, but also capture the local directional offsets between the candidate stroke and the reference stroke.

The last item in Eq. (5) calculates the cost caused by merging stroke segments. For a pair of adjacent stroke segments P_v^i, higher consistency score $s_{q\rightleftharpoons u}(P_v^i)$ should also cause smaller matching cost, which can be measured by $1 - s_{q\rightleftharpoons u}(P_v^i)$.

For the case that a reference stroke r_i is un-matched (matched with empty candidate stroke $N_{ij} = \varnothing$), the cost is calculated as

$$G(\varnothing) = \frac{1}{n_g}\sum_{k=1}^{n_g}\Big\{\Big[1 - MinDis(g_k, r_i)\Big] + C_s(g_k, r_i)\Big\}, \tag{7}$$

where n_g is the number of all the stroke segments in $CAND_i$. When $CAND_i$ is also empty, all stroke segments in M_c will be involved in the calculation of $G(\varnothing)$. Thus, the stroke segments with higher directional similarity or shorter distances to r_i cause larger un-match cost. The average calculation in Eq. (7) ensures that $G(\varnothing)$ locates in a comparable range with $G(N_{ij} \neq \varnothing)$.

4.2.4. Calculating $H(N_{ij})$

For N_{ij}, we need to use the heuristic function $H(N_{ij})$ in A^* search [49] to estimate the cost from N_{ij} to the goal node END. To calculate an admissible cost once reaching N_{ij}, we need to ensure that it never overestimates the actual cost to get to the goal node. Similar to the heuristic function in [14], $H(N_{ij})$ simply accumulates the minimum matching cost for each remaining reference stroke:

$$H(N_{ij}) = \sum_{k=i+1}^{N_R} min_m G(N_{km}), \tag{8}$$

where N_{km} denotes the m-th candidate node in $CAND_k$ of the reference stroke r_k.

To visualize the mechanism of the whole proposed stroke extraction process with stroke matching, we show an example in Fig. 8, where each node is represented as a reference-input stroke pair. In the complete path (depicted in red), each reference stroke is matched with a subset (including $\varnothing$) of stroke segments.

5. Experiments

5.1. Datasets and Performance Criteria

To evaluate the proposed stroke extraction method and involved tasks on offline handwritten character images, we need to collect enough annotated data. We use the CASIA-OLHWDB1.0 & 1.1 [32] datasets of isolated online handwritten Chinese characters (3,755 classes) to synthesize training data for our models and employ the ICDAR-2013 Online HCCR Competition dataset (ICDAR-Online) [42] to generate test data. This dataset contains freely-written characters, and many classes are beyond the general teaching syllabus in elementary educational. So, from CASIA-OLHWDB1.0 & 1.1 [32] and ICDAR-Online datasets [42], we collect a subset of 1,940 classes of frequently-used characters, which can meet the demands in primary education [4].

Besides, we collect Regularly Written characters from 20 different writers and select the same 1940 classes as CASIA-OLHWDB1.0 & 1.1 [32] and ICDAR-Online datasets [42]. We name this dataset as RW-OLHWDB and release it[1] to serve as one of our test datasets. Overall, all the datasets used in our paper are listed in Table 2, where all the samples are generated by the techniques in Fig. 3.

The trained models are applicable to offline character images directly, but because of the lacking of real offline data with stroke-level annotation, we evaluate the performance of stroke extraction

[1] RW-OLHWDB (Version 5), DOI10.5281/zenodo.4527700, https://zenodo.org/record/4527700

Table 2
Training/test datasets in our experiments.

Datasets for Training					
No.	Dataset	handwriting type	#writers	#samples (k)	#classe s
1	OLHWDB1.0 [32]	online	420	~814.8	1,940
2	OLHWDB1.1 [32]	online	300	~582.0	1,940
		Datasets for Test			
No.	Dataset	handwriting type	#writers	#samples (k)	#classes
3	ICDAR13-Online [42]	online	60	~116.4	1,940
4	RW-OLHWDB	online	20	~38.8	1,940

Table 3
Model Sizes and Training Settings in our experiments.

Model Sizes (MB)				
Backbone	SkeNet Branch	CrossNet Branch	PathNet	Overall
12.77	0.62	0.62	5.92	19.93
		Training Settings for SkeNet/CrossNet/PathNet		
Optimizer	No. of Epochs	Learning Rate (LR)	Weight Decay	Momentum
SGD	20	1×10^{-3}	2×10^{-4}	0.9
Batch Size	*LR* Policy	*LR Gamma*	GPUs	Platform
64	step (per 5 epochs)	0.1	2 × GeForce RTX 2070	TF1.0.0

on synthesized offline data from online handwritten samples. We will show some examples of stroke extraction from real offline images, however.

The performance criteria are as follows:

- For the skeletonization and cross region detection tasks, we use the pixel-level precision (P), recall (R) and F-measure to evaluate our models like [1,17,31]. Like the pixel-level evaluation of the similar foreground pixel detection method in [50], we treat all the predicted pixels located in the 4-neighborhoods of ground-truthed pixels as correct predictions.
- For the stroke extraction task, we measure the stroke-level precision (P^s), recall (R^s) and F^s-measure on the open-source evaluation system proposed by Kim et al. [1]: if one stroke is extracted successfully without any missed or redundant stroke segment, it is judged to be correct.

5.2. Architectures of Models

Our models, including the CrossNet f_c [1,2], the SkeNet f_s [17], and the PathNet f_p, where f_c and f_s share the same architecture [17] as shown in Fig. 7. The backbone networks of f_c, f_s are the convolutional part of the HCCRCNN9Layer [6], which is an effective deep model for handwritten Chinese character recognition (HCCR). Because our models in this paper should share the same inputs with their corresponding HCCR model [17,31], we use the parameters in the HCCRCNN9Layer to initialize f_c and f_s. The PathNet shares the same architecture with the VDSR-Net in [1,44], and is directly trained without pre-training. The model sizes with detailed training settings are listed in Table 3, where all the models need only 19.93MB parameters storage, and thus, are highly efficient.

5.3. Experimental Results

We give the results of the key steps in our stroke extraction system: cross detection, skeletonization, stroke extraction without template matching, and stroke extraction with template matching. Though the results of cross detection and skeletonization have bee reported in our previous paper [2,17], due to the re-implementation of CrossNet and SkeNet and the re-selection of test data, we give the new results in this paper for completeness.

Table 4
Results of cross detection.

models	ICDAR13-Online			RW-OLHWDB		
	P	R	F-measure	P	R	F-measure
OverlapNet [1]	0.762	0.704	0.731	0.793	0.735	0.763
HED Net [45]	0.849	0.786	0.816	0.890	0.825	0.856
DeepSke Net [31]	0.867	0.801	0.832	0.895	0.830	0.861
CrossNet (ours)	**0.869**	**0.803**	**0.834**	**0.904**	**0.838**	**0.869**

5.3.1. Cross Detection

In this task, we conduct experiments on four representative models with different architectures: OverlapNet [1] simply copied the architecture of VDSR-Net in [44] and was firstly deployed on the same task on printed Chinese characters (in *Standard Kaiti Font* [43]); HED Net [45] is the first deep FCN model for contour detection, which had a similar architecture with our SkeNet but runs without the inter-channel re-grouping [31] operation and multi-rate fusion [46]; DeepSke Net mainly adds the inter-channel regrouping module into HED Net; While based on DeepSke Net, our CrossNet uses a multi-rate convolutional fusion layer [46] to generate the final output [17]. For implementing the comparison methods (OverlapNet, HEDNet, DeepSke Net), we used the open source codes[2] and trained the models on the same training data with our model CrossNet.

The results of cross detection are shown in Table 4. The OverlapNet only stacks several convolutional layers successively without any task-oriented designs [1], so, it misses more pixels in cross regions than others. Comparing with HED Net and DeepSke Net, the inter-channel regrouping of feature maps in our model handles the multi-scale patterns (strokes) better. Furthermore, our CrossNet proves that the multi-rate dilated fusion [46] finds more details at the cross regions in handwritten character images. The CrossNet follows the architecture of the priorly proposed SkeNet [17] and shares the same backbone with the SkeNet. So, the CrossNet only

[2] OverlapNet: https://github.com/byungsook/vectornet/blob/master/models.py ; HEDNet: https://github.com/s9xie/hed/blob/master/examples/hed/train_val.prototxt ; DeepSke Net: https://github.com/zeakey/DeepSkeleton/blob/master/examples/DeepSkeleton/train_val.prototxt

Table 5
Results of skeletonization. $P_s/R_s/F_s$ denote the performance of SkeNet f_c, while $P_T/R_T/F_T$ denote the performance of S_T by combining the results of CrossNet and SkeNet.

methods	ICDAR13-Online						RW-OLHWDB					
	P_s	R_s	F_s-measure	P_T	R_T	F_T-measure	P_s	R_s	F_s-measure	P_T	R_T	F_T-measure
Stroke Correction [51]	0.322	0.348	0.335	-	-	-	0.394	0.410	0.402	-	-	-
Stroke Continuity [30]	0.448	0.484	0.465	-	-	-	0.485	0.502	0.493	-	-	-
ZhangSuen Thinning Algorithm [25]	0.456	0.492	0.473	-	-	-	0.523	0.545	0.533	-	-	-
HED Net [45]	0.693	0.712	0.702	0.705	0.724	0.714	0.768	0.790	0.778	0.771	0.793	0.781
DeepSke Net [31]	0.725	0.741	0.732	0.730	0.746	0.737	0.795	0.811	0.802	0.798	0.814	0.805
SkeNet	**0.747**	**0.769**	**0.757**	**0.759**	**0.782**	**0.770**	**0.811**	**0.821**	**0.815**	**0.819**	**0.830**	**0.822**

Table 6
Results of the proposed stroke extraction method **without** stroke matching. When $p_q \notin S_T$, we randomly choose p_q. $N_{q/u}$ denotes the number of query pixels. Potrace [52] treats each connected component as a standalone stroke.

methods	$p_q \in S_T$	have p_u	$N_{q/u}$	$D(p_q, p_u)$	ICDAR13-Online			RW-OLHWDB		
					P^s	R^s	F^s	P^s	R^s	F^s
Potrace [1,52]	-	-	-	-	0.283	0.221	0.248	0.420	0.357	0.385
OverlapNet+PathNet	×	×	1	-	0.734	0.681	0.706	0.802	0.780	0.790
	√	×	1	-	0.774	0.717	0.744	0.814	0.791	0.802
ours	√	√	1	4	0.823	0.784	0.803	0.874	0.851	0.862
	√	√	1	8	0.829	0.790	0.809	0.890	0.866	0.877
	√	√	1	12	0.810	0.772	0.790	0.875	0.851	0.862
	√	√	2	8	**0.841**	**0.802**	**0.821**	**0.906**	**0.882**	**0.893**
	√	√	3	8	0.797	0.759	0.778	0.852	0.829	0.840

incurs slight additional computation though its performance improvement over the DeepSke Net is only marginal.

Obviously, in Table 4, the precision P is always higher than the recall rate R, which indicates that there are still a considerable quantity of undetected pixels. But this issue has little influence on stroke extraction, because the goal of cross detection is to locate the intersection/touching points of different strokes, the detected pixels in cross regions are used to disconnect the stroke segments adjacent to them. Occasional missing or false positive cross region pixels do not cause serious error of stroke extraction, because stroke segments undergo dynamic merging in the following steps of stroke extraction and matching.

5.3.2. Skeletonization

In skeletonization task, we compare six representative methods, of which Stroke Correction [51], Stroke Continuity [30] and ZhangSuen Thinning Algorithm [25] are rule-based methods without training. We implemented them using the open source codes[3]. The results are shown in Table 5. In respect of precision, the FCNs report much better results than traditional methods (stroke correction, stroke continuity, and ZhangSuen). Benefiting from the multi-rate fusion [46], our model SkeNet outperforms the DeepSke Net considerably.

We can see that by combining the results of CrossNet and SkeNet, the performance of skeletonization on S_T is improved significantly over SkeNet f_s alone. This is because S_T improves the thinning results at cross regions particularly. From Table 4 & 5, we can see that all the methods report better performances on RW-OLHWDB because the more regularly written samples are easier to be processed.

Some examples of S_T combining SkeNet and CrossNet are shown in Fig. 10, where the bright pixels show the skeleton-wise cross regions, whose adjacent stroke segments need to be isolated or merged in stroke extraction. We can see that the CrossNet can capture almost all valid pixels, including the intersection pixels, which help refine the skeleton and split stroke segments.

[3] Stroke Correction and Stroke Continuity: https://pypi.org/project/thinning/ ; ZhangSuen Thinning: https://scikit-image.org/docs/dev/api/skimage.morphology.html#skimage.morphology.skeletonize

5.3.3. Stroke Extraction without Matching

For PathNet, the training target under a given query pixel p_q is the stroke image s_{p_q} which contains p_q. Therefore, we evaluate the proposed query pixel guided stroke extraction method on different selections of p_q in Table 6, where $D(p_q, p_u)$ denotes the distance between p_q and p_u shown in Fig. 11. As for the comparison methods, Potrace was implemented using the codes provided in [52]; for OverlapNet+PathNet, the OverlapNet was used for skeletonization and cross detection, and then the PathNet was used for stroke extraction. All the models were trained with the same data generated in our experiments.

The first row in Table 6 gives the results of Potrace [52], which is also the baseline method on the same task in [1]. Though Potrace simply regards each connected component as a standalone stroke, it can reveal the challenge of stroke extraction on our datasets. We can see that on the regularly written Chinese character dataset RW-OLHWDB, about 35.7% ground-truthed strokes are isolated components without intersection with others. But on the more challenging ICDAR13-online dataset, this rate drops to about 22.1%. For reference, when processing the printed Chinese characters in [1,37,43], Potrace can extract about 57% strokes because printed Chinese characters have more isolated strokes.

In Table 6, the rows of "OverlapNet+PathNet" give the results of a pure PathNet with one query pixel (i.e., the state-of-the-art method proposed by [1]). The results show the advantage of choosing p_q from S_T. This method uses only uni-directional query pixel guidance, thus the accuracy of stroke segment consistency is insufficient.

The last five rows of Table 6 give the results of our method (stroke extraction with bi-directional pixel guidance) with variable settings of $N_{q/u}$ and $D(p_q, p_u)$. First, when $N_{q/u} = 1$ (one query pixel, bi-directional guidance), the results of different $D(p_q, p_u)$ show that the selection $D(p_q, p_u) = 8$ yields good performance of stroke extraction. Then by fixing $D(p_q, p_u) = 8$, we vary $N_{q/u}$, and find that $N_{q/u} = 2$ yields superior performance of stroke extraction. Using more query pixels ($N_{q/u} = 3$) for measuring the consistency score of stroke segments, however, deteriorates the performance. This is because we select query pixels sequentially starting from the cross region with distance interval 4, the first and second query pixels are in moderate distance from the cross region, while more query pixels farther away from cross region are less reliable.

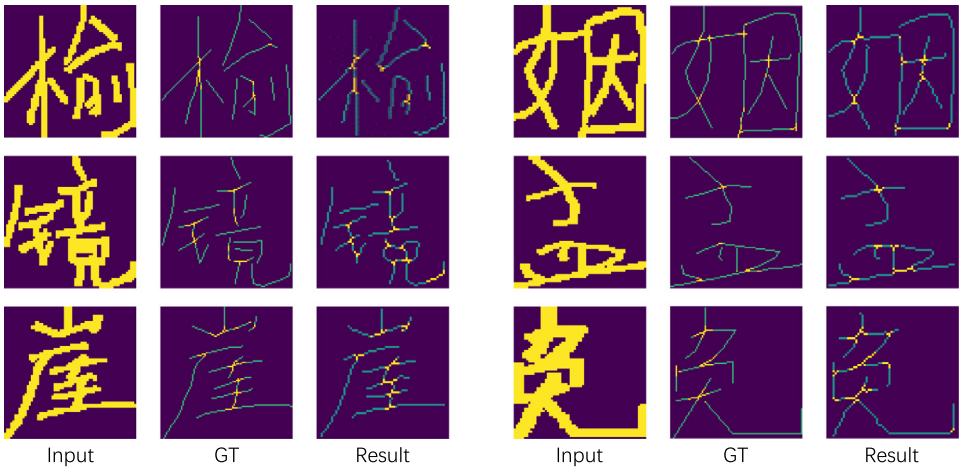

Fig. 10. Examples of skeletonzation combining the results of SkeNet and CrossNet. The input images, ground-truthed images and our results are listed from left to right. The dark pixels show all the stroke segments, while the bright pixels show the ground-truthed/detected cross regions.

Table 7
Results of the proposed stroke extraction method with stroke matching.

methods	$p_q \in S_T$	have p_u	$N_{q/u}$	$D(p_q, p_u)$	ICDAR13-Online			RW-OLHWDB		
					P^s	R^s	F^s	P^s	R^s	F^s
ours	√	√	1	4	0.868	0.853	0.860	0.895	0.880	0.887
			1	8	0.875	0.860	0.867	0.920	0.905	0.912
			1	12	0.886	0.871	0.878	0.925	0.910	0.865
			2	8	**0.890**	**0.875**	**0.882**	**0.949**	**0.934**	**0.941**
			3	8	0.859	0.844	0.851	0.889	0.875	0.881

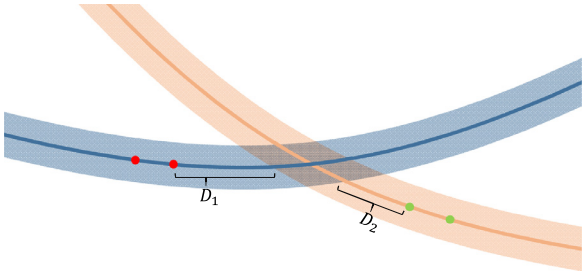

Fig. 11. $D(p_q, p_u) = D_1 + D_2$, where D_1 or D_2 measures the minimum distance from the representative points to their corresponding cross region. $D_1 = D_2$.

Table 8
Results of stroke extraction with matching on Chinese characters of different stroke number N_S.

N_S	#classes	ICDAR13-Online			RW-OLHWDB		
		P^s	R^s	F^s-measure	P^s	R^s	F^s-measure
<5	74	0.925	0.908	0.916	0.973	0.958	0.965
5	138	0.919	0.903	0.910	0.977	0.961	0.968
6	168	0.906	0.889	0.897	0.963	0.944	0.953
7	334	0.903	0.887	0.894	0.955	0.939	0.946
8	443	0.900	0.885	0.892	0.950	0.935	0.942
9	425	0.894	0.879	0.886	0.947	0.932	0.939
10	284	0.884	0.866	0.874	0.926	0.910	0.917
11	29	0.860	0.845	0.852	0.895	0.881	0.887
12	12	0.850	0.835	0.842	0.884	0.870	0.876
13	3	0.833	0.818	0.825	0.840	0.827	0.833
2∼13	1940	0.890	0.875	0.882	0.949	0.934	0.941

5.3.4. Stroke Extraction with Matching

Table 7 gives the results of stroke extraction with character model matching, at different options of $N_{q/u}$ and $D(p_q, p_u)$. Again, the choices $N_{q/u} = 2$ and $D(p_q, p_u) = 8$ yields the best performance. Compared to the results of stroke extraction without matching in Table 6, the stroke extraction precision and recall rates are improved significantly. Specifically, the best precisions on two datasets are improved by 4.9% and 4.3%, respectively. This verifies that the character templates play a very important role in disambiguating the stroke segment connection at cross regions.

As the complexity of stroke extraction generally increases with the stroke number, to verify this dependency, we also show the results of the proposed stroke extraction method on Chinese characters with different numbers of standard strokes N_S in Table 8. We can see that with increasing N_S, the performances of stroke extraction on two datasets decrease gradually. The stroke number 8 has the maximum number of character classes and also gets stroke extraction performance close to the average performance. Simple characters (with stroke number < 8) are main objects of elementary school learning. The high accuracies of stroke extraction on simple characters demonstrate the potential of the proposed method in application of character writing education.

5.3.5. Illustrative Examples

Fig. 12 and Fig. 13 show some examples of stroke extraction with matching on samples from ICDAR-Online dataset and RW-OLHWDB dataset, respectively. In the figures, “GT” denotes the ground-truth of strokes, “Ours-(a)” denotes the results of stroke extraction with matching; “Ours-(b)” denote the results of stroke extraction without matching; “Potrace” denotes the method [1,52] which treats connected components of skeleton as strokes. We can see that the Potrace method mis-extracts multiple intersected strokes as one stroke. The proposed method with template matching produces less stroke errors compared to that of stroke extraction without matching.

The handwritten characters in Figs. 12 & 13 cover most of the difficult cases in stroke extraction task, such as multi-stroke structures, multi-segment and curved strokes, variable stroke width,

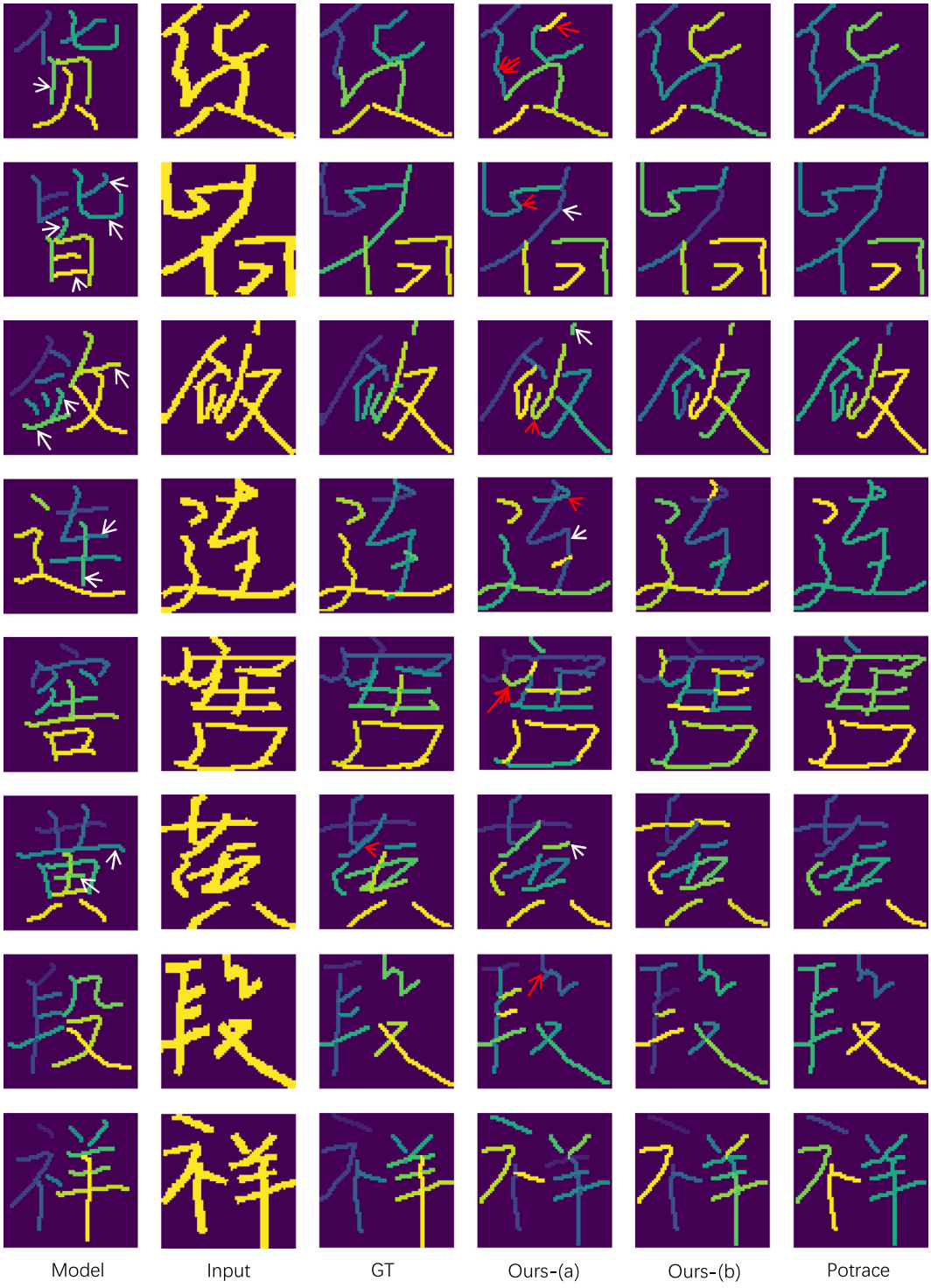

Fig. 12. Some samples randomly chosen from ICDAR13-Online dataset. Ours-(a): stroke extraction with matching; Ours-(b): stroke extraction without matching. The eight characters have 8(6), 9(7), 11(10), 7(8), 12(13), 11(10), 9(8) and 10(10) strokes, respectively. Here, "$m(n)$" means the reference character contains m strokes while the input handwritten character contains n strokes. Different colors indicate different strokes. The white arrows point to unmatched strokes, and the red arrows show those **un-extracted** and **mis-extracted** strokes.

and various shapes of stroke intersection. Many strokes are hard to identify even by human experts. The cursively written characters (in ICDAR13-Online dataset) produce more errors of missed or mis-extracted strokes. Short strokes are more likely to be missed or mis-matched because they are high variable in orientation.

The above results were obtained by evaluating on generated offline images synthesized from online handwritten samples. Though the proposed method, with models trained on generated data, are readily applicable to real offline images, it is hard to annotate a large number of offline character images at stroke level for performance evaluation. Nevertheless, to demonstrate the applicability, we show some examples of stroke extraction on real handwritten character images from the ICDAR13-Offline dataset [42] in Fig. 14. We can see the real offline characters have very similar appear-

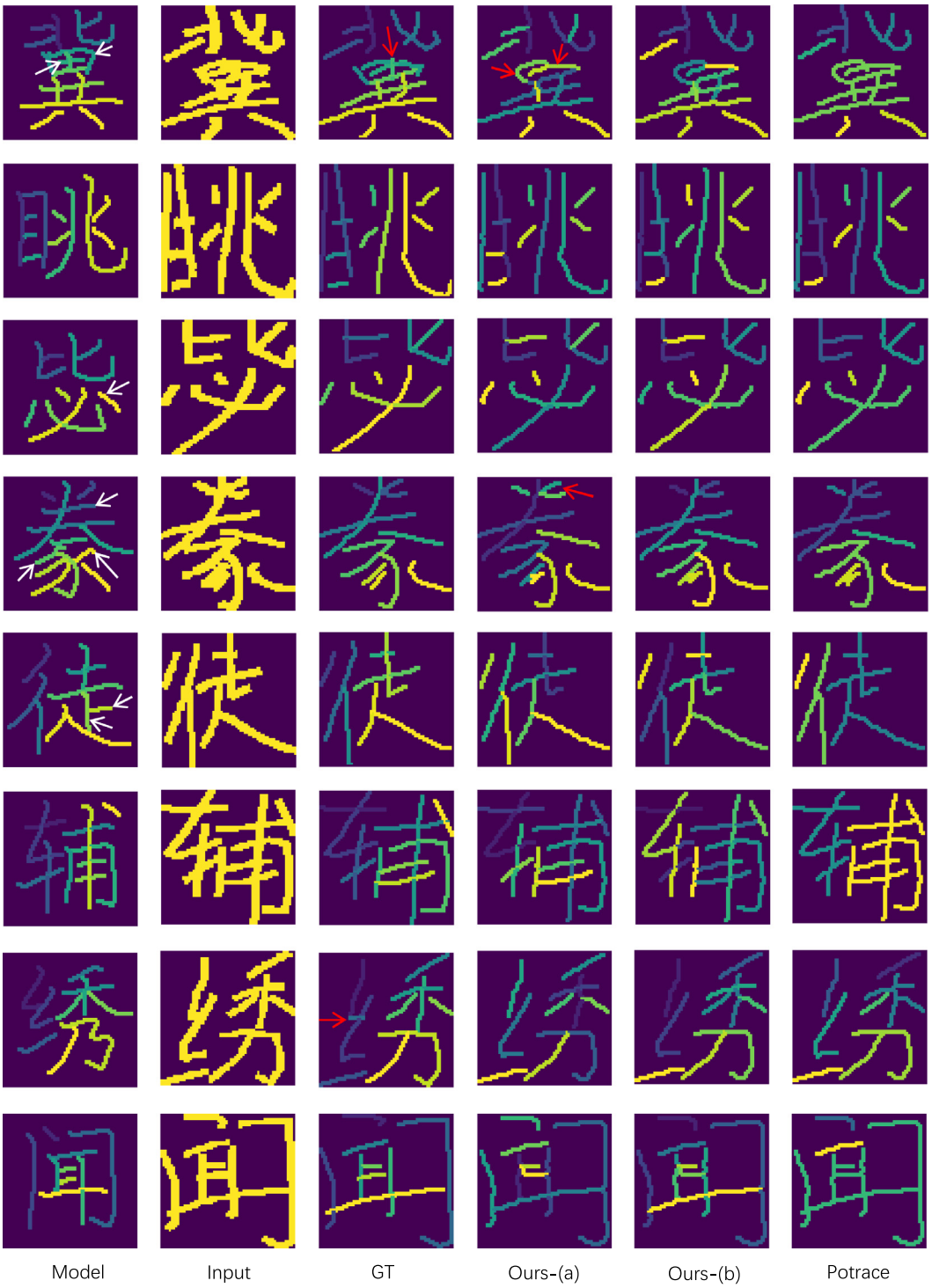

Fig. 13. Some samples randomly chosen from RW-OLHWDB dataset. The eight characters have 16(13), 11(11), 9(8), 13(10), 10(8), 11(11), 10(11) and 9(9) strokes respectively.

ance to generated images in Figs. 12 & 13, and the proposed stroke extraction method perform on offline characters as well as on generated images.

Finally, we show some typical errors of stroke extraction in Fig. 15. The major factors causing stroke extraction errors are as follows:

- A connected component with densely intersected multiple strokes is likely to make some strokes missed or merged with other strokes in skeletonization, such as the 1st sample in Fig. 15.
- Unduly distorted strokes are likely to have large dissimilarity with the standard strokes in reference models, so that they are failed to be matched or split into multiple strokes at cross/turning point. An example is shown in the 2nd sample of Fig. 15.
- Very short strokes or segments are likely to be missed in skeletonization or stroke matching, such as the 3rd sample in Fig. 15.

In general, cursive handwriting with unduly distorted stroke shape is likely to produce stroke extraction errors. For cursively

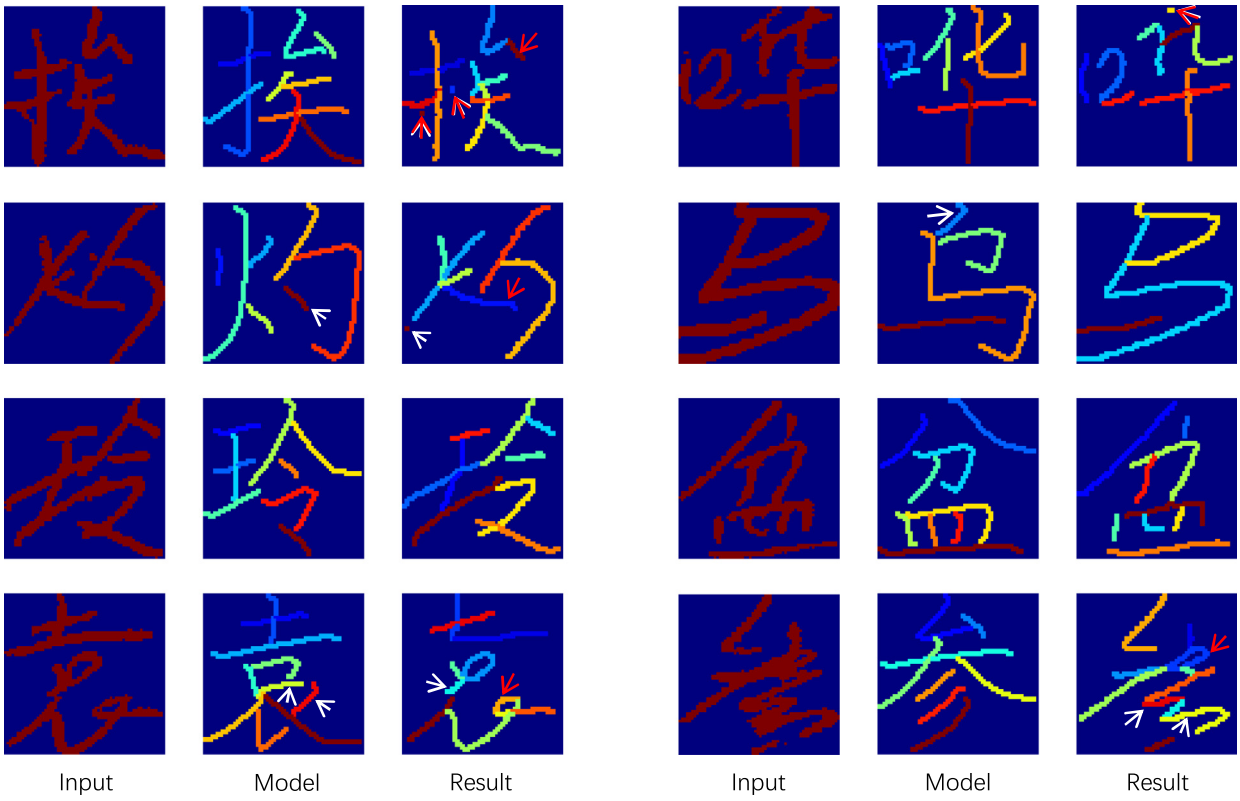

Fig. 14. Some examples of stroke extraction on real offline handwritten samples. From left to right: input character image, reference model, and strokes extracted. The write and red arrows point to the unmatched strokes and mis-extracted ones, respectively.

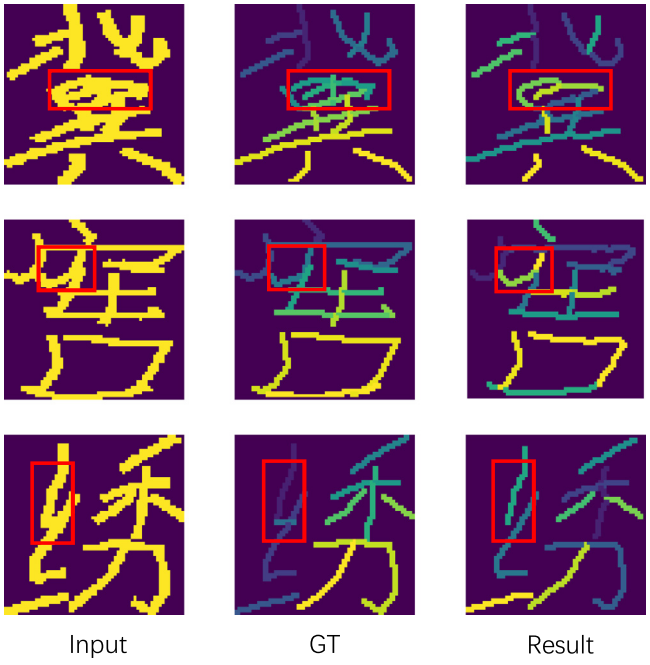

Fig. 15. Some examples of stroke extraction errors on our test datasets. From left to right: input character image, ground-truthed strokes, and strokes extracted. The regions of stroke extraction failure are boxed in red.

written characters, it is even hard for human experts to identify the correct strokes. In contrast, for printed Chinese character, it was shown in [1] that a pure PathNet without bi-directional guidance can reach ≥ 0.98 for both P^s and R^s.

6. Conclusion

In this paper, we proposed an effective method combining query pixel guidance and model-based stroke matching to address the challenging problem of stroke extraction for offline handwritten Chinese characters. In our system, the CrossNet detects cross regions (intersections of strokes), which bridge the skeleton segments into strokes. The detected cross regions can also improve the skeleton produced by the proposed SkeNet. Stroke extraction is performed by connecting skeleton-wise stroke segments adjacent to a cross region, and the pairing consistency between stroke segments is measured using a PathNet with bi-directional query pixel guidance. And finally, stroke matching with reference character models is performed by A* search to reduce the ambiguity of stroke connection.

The proposed method is readily applicable to offline handwritten characters, and we built synthetic datasets (with stroke-level ground-truths) to evaluate the performance convincingly. Our experiments report promising performances on skeletonization, cross detection, and stroke extraction. For potential applications, the proposed method can be applied to detect stroke-level irregularities, omissions, and redundancies in handwritten characters, and facilitate stroke-level writing quality assessment. Since the challenging problem of stroke extraction was not pursued intensively in recent years, we could not compare our results with other state-of-the-art results, but hopefully provide a benchmark for future studies. In our future work, better distance metrics are to be explored for dealing with high distortion in stroke matching. The end-to-end training of SkeNet, CrossNet and PathNet, and alternative design of stroke segment pairing consistency model also deserve attention in the future.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank *ARPHIC Company* and *Mr. Shaunak Kishore* for releasing their valuable data to public. This work was partly supported by the National Natural Science Foundation of China under Grants 61836014 and 61721004, and the EU Horizon 2020 RISE Project ULTRACEPT under Grant 778062.

References

[1] B. Kim, O. Wang, A.C. Öztireli, M. Gross, Semantic segmentation for line drawing vectorization using neural networks, Computer Graphics Forum 37 (2) (2018) 329–338.
[2] T.-Q. Wang, C.-L. Liu, DeepAD: A deep learning based approach to stroke-level abnormality detection in handwritten Chinese character recognition, in: IEEE International Conference on Data Mining, 2018, pp. 1302–1307.
[3] Arphic, (http://www.arphic.com.tw/en/).
[4] The Basics of Chinese characters, (https://www.thoughtco.com/basics-about-chinese-characters-4080664).
[5] X.-Y. Zhang, Y. Bengio, C.-L. Liu, New benchmark for online and offline handwritten chinese character recognition with deep convolutional network and adaptation, Pattern Recognition 61 (2017) 348–360.
[6] X. Xiao, L. Jin, Y. Yang, W. Yang, J. Sun, T. Chang, Building fast and compact convolutional neural networks for offline handwritten Chinese character recognition, Pattern Recognition 72 (2017) 72–81.
[7] T.-B. Xu, P. Yang, X.-Y. Zhang, C.-L. Liu, Lightweightnet: Toward fast and lightweight convolutional neural networks via architecture distillation, Pattern Recognition 88 (2019) 272–284.
[8] Z. Cao, J. Lu, S. Cui, C. Zhang, Zero-shot handwritten chinese character recognition with hierarchical decomposition embedding, Pattern Recognition 107 (2020) 107488.
[9] J. Zhang, J. Du, L. Dai, Radical analysis network for learning hierarchies of chinese characters, Pattern Recognition 103 (2020) 107305.
[10] L.Y. Tseng, C.-T. Chuang, An efficient knowledge-based stroke extraction method for multi-font Chinese characters, Pattern Recognition 25 (12) (1992) 1445–1458.
[11] C. Lee, B. Wu, A Chinese-character-stroke-extraction algorithm based on contour information, Pattern Recognition 31 (6) (1998) 651–663.
[12] H. Ogawa, K. Taniguchi, Thinning and stroke segmentation for handwritten Chinese character recognition, Pattern Recognition 15 (4) (1982) 299–308.
[13] K. Liu, Y.S. Huang, C.Y. Suen, Robust stroke segmentation method for handwritten Chinese character recognition, in: International Conference on Document Analysis and Recognition, volume 1, 1997, pp. 211–215.
[14] C.-L. Liu, I.-J. Kim, J.H. Kim, Model-based stroke extraction and matching for handwritten Chinese character recognition, Pattern Recognition 34 (12) (2001) 2339–2352.
[15] F. Lin, X. Tang, Off-line handwritten Chinese character stroke extraction, in: International Conference on Pattern Recognition, 2002, pp. 249–252.
[16] J. Tan, J. Lai, W.-S. Zheng, C.Y. Suen, A novel approach for stroke extraction of off-line Chinese handwritten characters based on optimum paths, in: International Conference on Frontiers in Handwriting Recognition, 2012, pp. 786–790.
[17] T.-Q. Wang, C.-L. Liu, Fully convolutional network based skeletonization for handwritten Chinese characters, in: AAAI Conference on Artificial Intelligence, 2018, pp. 2540–2547.
[18] K. Liu, Y.S. Huang, C.Y. Suen, Identification of fork points on the skeletons of handwritten Chinese characters, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (10) (1999) 1095–1100.
[19] C.-C. Hsieh, H.-J. Lee, A probabilistic stroke-based Viterbi algorithm for handwritten Chinese characters recognition, in: International Conference on Pattern Recognition, 1992, pp. 191–194.
[20] C.-C. Hsieh, H.-J. Lee, Off-line recognition of handwritten Chinese characters by on-line model-guided matching, Pattern Recognition 25 (11) (1992) 1337–1352.
[21] F.-H. Cheng, Multi-stroke relaxation matching method for handwritten chinese character recognition, Pattern Recognition 31 (4) (1998) 401–410.
[22] J. Rocha, T. Pavlidis, A shape analysis model with applications to a character recognition system, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (4) (1994) 393–404.
[23] List of CJK fonts, (https://en.wikipedia.org/wiki/List_of_CJK_fonts).
[24] Y. Qiao, M. Nishiara, M. Yasuhara, A framework toward restoration of writing order from single-stroked handwriting image, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006) 1724–1737.
[25] T. Zhang, C.Y. Suen, A fast parallel algorithm for thinning digital patterns, Communications of the ACM 27 (3) (1984) 236–239.
[26] C. Arcelli, G.S. Di Baja, A width-independent fast thinning algorithm, IEEE Transactions on Pattern Analysis and Machine Intelligence 7 (4) (1985) 463–474.
[27] M.A. Alghamdi, W.J. Teahan, A new thinning algorithm for Arabic script, International Journal of Computer Science and Information Security 15 (1) (2017) 204.
[28] C. Arcelli, G.S. Di Baja, A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (4) (1989) 411–414.
[29] J.J. Zou, H. Yan, Skeletonization of ribbon-like shapes based on regularity and singularity analyses, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 31 (3) (2001) 401–407.
[30] J. Dong, Y. Chen, Z. Yang, B.W.-K. Ling, A parallel thinning algorithm based on stroke continuity detection, Signal, Image and Video Processing 11 (5) (2017) 873–879.
[31] W. Shen, K. Zhao, Y. Jiang, Y. Wang, Z. Zhang, X. Bai, Object skeleton extraction in natural images by fusing scale-associated deep side outputs, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 222–230.
[32] C.-L. Liu, F. Yin, D.-H. Wang, Q.-F. Wang, CASIA online and offline Chinese handwriting databases, in: International Conference on Document Analysis and Recognition, 2011, pp. 37–41.
[33] Z. Zhou, E. Zhan, J. Zheng, Stroke extraction of handwritten Chinese character based on ambiguous zone information, in: International Conference on Multimedia and Image Processing, 2017, pp. 68–72.
[34] R. He, H. Yan, Stroke extraction as pre-processing step to improve thinning results of Chinese characters, Pattern Recognition Letters 21 (9) (2000) 817–825.
[35] L. Sun, M. Liu, J. Hu, X. Liang, A Chinese character teaching system using structure theory and morphing technology, PLoS ONE 9 (6) (2014) e100987.
[36] X. Chen, Z. Lian, Y. Tang, J. Xiao, An automatic stroke extraction method using manifold learning, in: Annual Conference of the of the European Association for Computer Graphics, 2017, p. 6568.
[37] Make Me a Hanzi, (https://www.skishore.me/makemeahanzi/).
[38] C.H. Leung, Y.S. Cheung, Y.L. Wong, A knowledge-based stroke-matching method for Chinese character recognition, IEEE Transactions on Systems, Man, and Cybernetics 17 (6) (1987) 993–1003.
[39] L.-H. Chen, J.-R. Lieh, Handwritten character recognition using a 2-layer random graph model by relaxation matching, Pattern Recognition 23 (11) (1990) 1189–1205.
[40] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering 22 (10) (2010) 1345–1359.
[41] X.-Y. Zhang, F. Yin, Y.-M. Zhang, C.-L. Liu, Y. Bengio, Drawing and recognizing Chinese characters with recurrent neural network, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (4) (2018) 849–862.
[42] F. Yin, Q.-F. Wang, X.-Y. Zhang, C.-L. Liu, ICDAR 2013 Chinese handwriting recognition competition, in: International Conference on Document Analysis and Recognition, 2013, pp. 1464–1470.
[43] hanzi-writer-data CDN files, (https://cdn.jsdelivr.net/npm/hanzi-writer-data@2.0/).
[44] J. Kim, J. Kwon Lee, K. Mu Lee, Accurate image super-resolution using very deep convolutional networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1646–1654.
[45] S. Xie, Z. Tu, Holistically-nested edge detection, in: IEEE International Conference on Computer Vision, 2015, pp. 1395–1403.
[46] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, G. Cottrell, Understanding convolution for semantic segmentation, in: IEEE Winter Conference on Applications of Computer Vision, 2018, pp. 1451–1460.
[47] A. Rosenfeld, E. Johnston, Angle detection on digital curves, IEEE Transactions on Computers 22 (1973) 875–878.
[48] B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence 129 (1) (2001) 5–33.
[49] M. Nosrati, R. Karimi, H.A. Hasanvand, Investigation of the *(star) search algorithms: Characteristics, methods and approaches, World Applied Programming 2 (4) (2012) 251–256.
[50] W. Xiong, J. Yu, Z. Lin, J. Yang, X. Lu, C. Barnes, J. Luo, Foreground-aware image inpainting, in: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5840–5848.
[51] A.K. Pujari, C. Mitra, S. Mishra, A new parallel thinning algorithm with stroke correction for Odia characters, in: Advanced Computing, Networking and Informatics, volume 1, Springer, 2014, pp. 413–419.
[52] Potrace, http://potrace.sourceforge.net.

Tie-Qiang Wang received the B.S. degree in electronics science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2015. He is currently working toward the PhD degree in pattern recognition and intelligent systems at the Institute of Automation, Chinese Academy of Sciences, Beijing, China and the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China. His research interests include handwritten character recognition, document image analysis, and deep learning.

Xiaoyi Jiang received the bachelors degree from Peking University, Beijing, China, and the Ph.D. and Venia Docendi (Habilitation) degrees from the University of Bern, Bern, Switzerland, all in Computer Science. He was an Associate Professor with the Technical University of Berlin, Berlin, Germany. Since 2002, he has been a Full Professor with the University of Münster, Münster, Germany, where he is currently the Dean of the Faculty of Mathematics and Computer Science. His current research interests include biomedical imaging, 3-D image analysis, and structural pattern

recognition. Dr. Jiang is the Editor-in-Chief of the International Journal of Pattern Recognition and Artificial Intelligence. He also serves on the Advisory Board and the Editorial Board of several journals, including IEEE Transactions on Medical Imaging and International Journal of Neural Systems. He is a Senior Member of IEEE and a Fellow of IAPR.

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 director of the laboratory. He is also a deputy director of the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China. He received the B.S. degree in electronic engineering from Wuhan University, Wuhan, China, the M.E. degree in electronic engineering from Beijing University of Technology, Beijing, China, the Ph.D. degree in pattern recognition and intelligent control from the Institute of Automation of 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, neural networks, machine learning, and especially the applications to character recognition and document analysis. He has published over 300 technical papers at prestigious journals and conferences. He is an associate editor-in-chief of Pattern Recognition, and is on the editorial board of International Journal on Document Analysis and Recognition, Cognitive Computation and CAAI Trans. Intelligence Technology. He is a fellow of the CAA, the CAAI, the IEEE and the IAPR.