Pen Trace Reconstruction with Skeleton Representation of a Handwritten Text Image*

Svetlana Kryzhanovskaya, Sergey Arseev, Leonid Mestetskiy

Lomonosov Moscow State University, 119991 Leninskie Gory, Moscow, Russia
skryzhanovskaya@yandex.ru
9413serg@gmail.com
mestlm@mail.ru

Abstract. This paper presents a new approach for the pen trace reconstruction task in handwritten text images. The proposed approach is based on the skeleton representation of a binary image which provides sufficient information about the pen movement trajectory during the writing. As such, the initial problem can be viewed as a problem of constructing a specific walk in the skeleton graph covering all its edges. The proposed approach consists of splitting the graph into structural elements and estimating the pen movement direction in each of them with the resulting trace being composed of the traces of these elements.

Keywords: Pen Trace Reconstruction, Binary Image Skeleton, Stroke-based Decomposition, Handwritten Text Recognition.

1 Introduction

The pen trace reconstruction is the transformation of a given binary image containing a connected piece of handwritten text into a trace represented as a sequence of points where the pen is supposed to have been tracing over them consequently.

This reconstruction is a possible approach to the wider task of offline handwriting recognition where the text is recovered from an image of a piece of handwritten text. No universal solution for this task has yet been developed and the current state of the art solutions impose major restrictions on the input and usually require a large amount of training data. The other related task is the online handwriting recognition where the input is provided as a sequence of pen positions instead of an image. This representation usually has significantly lower input dimensionality than the image and is less sensitive to noise which allows this task to be solved more efficiently. However, it requires the use of a specific hardware for the input data capture which greatly lowers the range of its practical applications.

Therefore, the idea described in [7] is to reduce the offline recognition task to the online recognition task using the reconstructed pen trace from an image to simulate an online input.

* Supported by Russian Foundation for Basic Research, grant #20-01-00664.

The task of pen trace reconstruction in handwritten text images has a wide range of practical applications as a part of the handwriting recognition task. If the offline recognition task is successfully reduced to the online recognition task, more efficient and less restrictive methods can be used in tasks such as automated archive digitalization. Another example of the pen trace reconstruction applications is the task of text samples verification via comparing them to the reference. For example, in [8] the reconstructed trajectory is used for signature identification.

2 Related work

Many approaches exist for the given task, however they still don't provide the required accuracy in all cases. Most of them can be roughly split into three main groups: local, global and artificial neural network based methods.

Global methods attempt to reconstruct the pen trajectory by optimizing a global quality functional. For example, in [6] the total curvature of the trace is minimized and the task is reduced to solving a travelling salesman task on a skeleton graph. The main drawback of global methods is that they are computationally expensive because they involve optimization over a large set of possibile options.

In local methods, ambiguous zones of the trace are processed separately. In [8], a number of heuristic rules is proposed to solve these zones. In [3], a discrete transform of the skeleton is used with subsequent classification of its nodes. The main drawback of these methods is that they are often based on assumption that the pen trace is continuous.

Recently, there has been a surge in methods using artificial neural networks. In [14], convolutional neural networks (CNN) are used for pen trace reconstruction in Chinese symbols, and in [1], the use of recurrent neural networks (RNN) is proposed to recover the trajectory in Indian scripts. In [12] RNN End-To-End framework combines with CNN for feature extraction. However, most existing methods based on neural networks are only effective for trajectory reconstruction in isolated symbols and require a large amount of training data.

The main advantages of the proposed approach is that it is applicable for entire words. What is more, it combines local and global trajectory reconstruction methods and doesn't require labeled training dataset.

3 Proposed method

The main idea of the proposed method is that the binary image is transformed into a geometric graph where each node corresponds to the pen position at some point during writing and edges model the pen movement between nodes. This way, the initial problem is solved by consructing a sequence of walks covering all edges of the graph together. Since the handwriting can consist of separate strokes, the trace doesn't necessary consist of a single walk. What is more, the pen can trace some parts of its trajectory multiple times so it is possible that an edge will be included more than once and re-traced both in the same or opposite direction.

3.1 Skeleton representation

We use a skeleton of a binary image as a graph modeling the pen trajectory. The stages of its construction are:

1. Object border approximation with contour representation, i.e. a polygon with a minimum perimeter separating black and white pixels of the binary image.
2. Calculation of a skeleton of the contour representation as a locus of the centers of circles that are tangent to the polygon edge in two or more points, where all such circles are contained in the figure. The skeleton is then approximated as a geometric graph with the nodes corresponding to the centers of some of the circles and edges being straight lines.
3. Skeleton pruning, i.e. cutting off the shortest edges that don't affect the overall shape significantly. Without pruning, these branches add significant noise to the pen trace model.

The skeleton graph is described in greater detail in [10]. An example of a binary image with its skeleton graph can be seen on Fig. 1.

If the skeleton consists of multiple connected components, each one is traced separately. The tracing algorithm for one component is presented below.

Fig. 1. Contour representation (green line) and skeleton (red line) of a binary image.

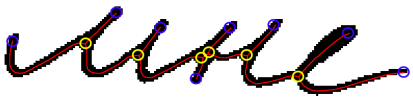

Fig. 2. Terminal nodes (in blue circles) and intersections (in yellow circles).

3.2 Stroke-based segmentation

Lets define a degree 1 node of a graph as a terminal node and a node with a degree higher than 2 as an intersection. Note that removing this nodes will split the graph into subgraphs that we call branches. Examples of terminal nodes and intersections can be seen on Fig. 2.

We assume that the pen can only lift from the paper in terminal nodes and intersections. A stroke is a subgraph of a skeleton graph that consists of one or more branches and corresponds to a continuous fragment of the pen trace. The result of a stroke-based segmentation is a set of strokes covering all edges of the graph. It is possible for some of the strokes to have common branches (usually, in the parts of the trajectory that are traced twice).

Cyclical strokes Cyclical strokes correspond to rounded graphic elements of handwritten letters. For each of the strokes, it is possible to trace it either clockwise or counter-clockwise. For example, in Russian language, cycles located below the baseline (baseline extraction methods are described in [11]) are usually traced clockwise and other cycles are traced counter-clockwise.

Each cyclical stroke is a cycle in the skeleton graph. To find all cycles in an undirected graph we use the depth-first search described in [2]. In the set of all cycles we select a subset that satisfies the following criteria:

1. All cycle edges are covered.
2. Total length of the intersections between cycles is minimized (minimize the amount of parts traced twice).

Vertical strokes Vertical strokes correspond to vertical graphic elements of handwritten letters. We assume that these strokes are traced from the top down. Each such stroke is a chain in the graph that begins and ends in a terminal node.

For a pair (u, v) of terminal nodes of the skeleton graph, the following symbols are introduced:

1. l is a line passing through u and v.
2. $s = (u, v_1, \ldots, v_n, v)$ is a shortest path from u to v in the graph.
3. $dist(l, v)$ is a distance from l to v.

The pair (u, v) satisfies the verticality condition if

1. $u.y < v.y$ (u – "bottom" node, v – "top" node);
2. The tilt of l is in range $[\alpha_1, \alpha_2]$, where α_1, α_2 are parameters of the algorithm.

The pair (u, v) satisfies the linearity condition if

$$\mathcal{L}(u, v) = \frac{1}{n} \sum_{j=1}^{n} dist(l, v_i) < \beta, \beta \text{ is a parameter}$$

The chains in the graph with both ends being terminal nodes satisfying the verticality and linearity conditions are designated as vertical strokes.

Vertical strokes cannot have common branches with each other.

Semi-vertical strokes Semi-vertical strokes are designated in a similar way as vertical strokes, but a semi-vertical stroke is a chain in a graph between the top terminal node and bottom intersection node.

Semi-vertical strokes are designated after vertical strokes. Semi-vertical strokes cannot have common branches with each other and vertical strokes.

Simple strokes Simple strokes are chains that consist of a single branch of the skeleton graph.

Simple strokes are designated after all other strokes have been found. A simple stroke cannot have common branches with other strokes.

Fig. 3. Skeleton graph decomposition into strokes: cyclical (blue), vertical (green), semi-vertical (yellow) and simple (red).

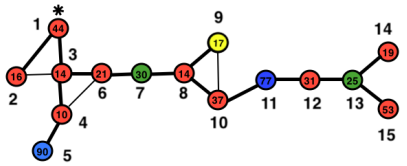

Fig. 4. Weighted metagraph and its root (marked with an asterisk), spanning tree (highlighted with bold lines) and corresponding sequence of tracing nodes (specified by numbers).

3.3 Stroke sequence synthesis

$\{S_1, \ldots, S_n\}$ is a set of strokes from the stroke-based segmentation. An example of a stroke-based segmentation can be seen on Fig. 3.

Metagraph construction A metagraph of a skeleton graph and its stroke decomposition is a weighted graph where the nodes correspond to the strokes of the initial graph and any two nodes are connected with an edge if their corresponding strokes have common nodes. The weight $W(S_i)$ of each node is a total length of the edges of the corresponding stroke.

For skeleton graph and its decomposition into strokes on Fig. 3 a corresponding metagraph can be seen on Fig. 4.

Metagraph spanning tree construction

1. Selection of a root stroke to be a starting node for a walk. Since in the Russian language words are written left to right, the stroke that contains a node with the minimum horizontal coortinate is selected as a starting stroke for the Russian language.
2. Width-first search in the metagraph, starting in the root stroke (described in [2]). The spanning tree consists of all pairs (u, v) where the algorithm processing the node u finds a new, not previously marked node v incidental to it.

On Fig. 4 the root is marked with an asterisk and the spanning tree is highlighted with bold lines.

Spanning tree walk For each node we calculate the weight of the "heaviest" subtree:

$$D(S_i) = \begin{cases} W(S_i) & \text{if } S_i \text{ is a leaf} \\ W(S_i) + \max\limits_{S_j \in children(S_i)} D(S_j) & \text{otherwise} \end{cases}$$

Algorithm 1 Spanning tree tracing algorithm

```
1: function METASEARCH( S_cur)
2:     if S_cur is not a leaf then
3:         sort S_i ∈ children(S_cur) by D(S_i) in ascending order
4:         for all S_i ∈ children(S_cur) in this order do
5:             METASEARCH( S_i)
6: METASEARCH(S_start)
```

On Fig. 4 the sequence of trace nodes obtained by the algorithm is specified by numbers.

Simple stroke direction Let $(S_{(1)}, \ldots, S_{(n)})$ be a sequence of strokes provided by the algorithm 1.

If $S_{(1)}$ is a simple stroke, it is traced from the leftmost end node.

Let $S_{(i)}$ be a simple node, $i > 1$. Find $S_{(j)}$, so that:

1. $j < i$
2. $S_{(j)} \cap S_{(i)} \neq \emptyset$
3. $\forall k \ : j < k < i, \ \ S_{(k)} \cap S_{(i)} = \emptyset$

Since each simple stroke is a branch, $S_{(i)} \cap S_{(j)} = \{v\}$, and v is an end node of $S_{(i)}$. This way$S_{(i)}$ is traced starting from v.

4 Experiments

For testing purposes and evaluation, the algorithm was implemented using the Python programming language. A library [9] was used to calculate the skeleton representation. For visual verification of the program, a graphical interface simulating the pen movement was developed.

4.1 Word level experiment

Dataset and experiment conditions The method was evaluated on a set of 250 labeled binary images depicting scanned handwritten words in the Russian language. Examples of the images in the testing set can be seen on Fig. 5. For each image, a skeleton was labeled by an assessor. The markup constsis of a sequence of the skeleton graph branches with their trace direction for the possible trajectory. Note that extra branches can appear in the skeleton not corresponding to the actual trajectory. In that case, these branches were not labeled and did not affect the quality metric.

Quality metrics Since it's assumed that the pen can only lift from the paper in intersections and terminal nodes, all reconstruction errors fall into two categories:

1. Branch trace direction errors
2. Branch ordering errors

We used the following quality metrics for evaluation:

1. The fraction of branches with the correctly chosen trace direction.
2. Levenshtein distance or the total amount of insertions, deletions, substitutions and transpositions required to turn the algorithm-produced branch ordering into the ground truth ordering performed by a human.

Experiment results A total of 5940 branches were marked for 250 images. Branch direction predicted by the algorithm matched the assessor labels for 5762 branches (97 %).

For branch ordering, the total Levenshtein distance to corresponding ground truth labels for all 250 samples equals 432 (average ≈ 2).

These results are viewed as positive. It's worth noting that in most cases several different handwriting styles are possible and as such, more than one trajectory can be considered to be the ground truth. In this experiment, only one possible trajectory was labeled by the assessor and the possibility for the algorithm to select another "true" trajectory can potentially lower the metrics in relation to the actual quality.

Table 1. Experiment results 1.

Fraction of correctly traced branches	**Levenshtein distance (average)**
0.97	≈ 2

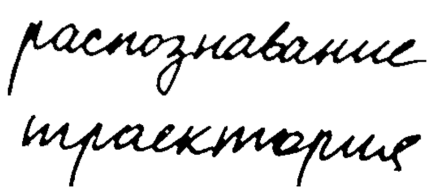

Fig. 5. Input images examples for word level experiment.

Fig. 6. Input images examples for letter level experiment.

4.2 Letter level experiment

Dataset and experiment conditions To test the ability of the proposed method to be used for online recognition, the method was evaluated on a set of 700 labeled binary images depicting various characters from English alphabet. It consists of seven classes of symbols: A, B, E, P, S, W and None, the first six containing data of the respective letter and the last containing various symbols not corresponding to any of the other classes. The dataset has 100 images for each class that were rasterized using traces gathered with the Low Power Projected Capacitive Touchpad Development Kit tool by Microchip. Some examples of the data are shown on Fig. 6.

Quality metrics The main quality criterion of the reconstructed pen trace for the recognition task is the comparison between offline recognition, online recognition with true pen trace and offline-to-online recognition with the reconstructed pen trace. An algorithm implemented in [5] was selected for experimental evaluation of the pen trace reconstruction algorithm. This method uses a recurrent neural network consisting of 24 gated recurrent units with sigmoid activation function. The model was trained from scratch; images were split into training and testing sets in 4:1 proportion. The training set has been enhanced using random shifts for sequence points. The offline recognition method selected for comparison was the VGG16 convolutional network described in [13] and pre-trained on the ImageNet dataset [4]. The training and testing sets split of the data has also been conducted in 4:1 proportion. Convolutional neural networks achieve good results but require a large amount of training data to learn properly. However, in practical applications it's not always possible to obtain a large marked data set relevant to the specific task. Different writing systems and different conditions require different data sets and it is often too tedious to perform a large scale data gathering and labeling for the task. The proposed method significantly reduces the complexity of the model and dimensionality of the data which enables it to produce practical results on a wide variety of different tasks without the need for long and costly data preparation.

The quality metric used for comparison was the proportion of correctly classified samples.

Experiment results The results of the evaluation are shown in the table. Recognition methods are marked as follows:

1. RNN-True – recurrent neural network from [5] trained on the real pen trace data.
2. RNN-Offline – same network trained on reconstructed pen trace data provided by the proposed algorithm.
3. VGG – the VGG16 convolutional network.

The proposed method shows lower classification accuracy than recognition based on true trajectories, however its results are significantly higher than the offline recognition method. The low performance of the VGG16 network can be attributed to low size of the training set which, despite the help of transfer learning and data augmentation, is not enough for the complex convolutional models while the proposed method shows higher performance despite having only 80 training images for each class.

Table 2. Experiment results 2.

Method	Accuracy
RNN-True	0.93
RNN-Offline	0.88
VGG	0.60

5 Conclusion

This work presents a new approach to the pen trace reconstruction by a handwritten text image using its skeleton representation. The algorithm was evaluated on a set of data to measure its performance.

Advantages of the proposed approach

1. The trajectory is not required to be continuous.
2. Some parts of the trajectory can be re-traced.
3. No labeled training dataset is required for word-level segmentation.
4. Can be used not only on isolated symbols but on continuous words as well.
5. When used for recognition, the method can succesfully learn on a very small labeled dataset which enables it to be used for tasks with scarce data where convolutional models cannot be trained properly.

Drawbacks of the proposed approach

1. Skeleton graph doesn't always accurately represent the pen trajectory. For example, it can contain extra branches not corresponding to any part of the trace.
2. Stroke-based segmentation uses heuristics derived from language specifics.

Further work in this area will include integration of the pen trace reconstruction method into online text recognition systems to enable solving the offline recognition problem using online recognition methods.

References

1. Bhunia, A.K., Bhowmick, A., Bhunia, A.K.: Handwriting trajectory recovery using end-to-end deep encoder-decoder network. ICPR (2018)
2. Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms (2007)
3. Crispo, G., Diaz, M., Marcelli, A., Ferrer, M.: Tracking the ballistic trajectory in complex and long handwritten signatures (08 2018). https://doi.org/10.1109/ICFHR-2018.2018.00068
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: Imagenet: a large-scale hierarchical image database. pp. 248–255 (06 2009). https://doi.org/10.1109/CVPR.2009.5206848
5. Fischer, T.: (2018), https://github.com/tobiasfshr/online-handwritten-character-recognition-capacitive-sensors
6. Jaeger, S.: Recovering writing traces in off-line handwriting recognition: Using a global optimization technique. vol. 3, pp. 150 – 154 vol.3 (1996)

7. Lallican, P.M., Viard-Gaudin, C., Knerr, S.: From off-line to on-line handwriting recognition (2004)
8. Lee, S., Pan, J.C.J.: Offline tracing and representation of signatures. IEEE Trans. Systems, Man, and Cybernetics **22**, 755 – 771 (1992)
9. Mestetskiy, L.: http://www.machinelearning.ru/wiki/images/d/da/SkeletonMaker_C.rar
10. Mestetskiy, L.: Continuous morphology of binary images: figures, skeletons, circulars. Moscow: Fizmatlit (2009)
11. Quirós, L.: Multi-task handwritten document layout analysis (06 2018)
12. Rabhi, B., Elbaati, A., Hamdi, Y., Alimi, A.M.: Handwriting recognition based on temporal order restored by the end-to-end system. In: 2019 International Conference on Document Analysis and Recognition (ICDAR). pp. 1231–1236 (Sep 2019). https://doi.org/10.1109/ICDAR.2019.00199
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR (2014)
14. Zhao, B., Yang, M., Tao, J.: Drawing order recovery for handwriting chinese characters. pp. 3227–3231 (2019)