Recognizing Off-line Flowcharts by Reconstructing Strokes and Using On-line Recognition Techniques

Martin Bresler[1], Daniel Průša[1], Václav Hlaváč[2]
Czech Technical University in Prague
Center for Machine Perception spanning the Faculty of Electrical Engineering[1]
and the Czech Institute of Informatics, Robotics and Cybernetics[2]
166 27, Praha 6, Technická 2, Czech Republic
{*breslmar, prusapa1*}*@fel.cvut.cz, hlavac@ciirc.cvut.cz*

***Abstract*—We experiment with off-line recognition of hand-written flowcharts based on strokes reconstruction and our state-of-the-art on-line diagram recognizer. A simple baseline algorithm for strokes reconstruction is presented and necessary modifications of the original recognizer are identified. We achieve very promising results on a flowcharts database created as an extension of our previously published on-line database.**

***Keywords*-flowcharts; strokes reconstruction; on-line recognition; off-line recognition;**

I. Introduction

Handwriting recognition can be divided into two basic approaches: on-line and off-line. We speak about the on-line recognition when the input is a sequence of strokes captured by an ink input device such as a tablet. The stroke is a sequence of points captured between pen-down and pen-up events. Every stroke point is defined by its coordinates on the planar drawing canvas. Additional data like a time stamp or a pressure value may be attached to it. We speak about the off-line recognition when the input is a raster image. The on-line recognition reaches naturally higher precisions, because of the additional dynamic information. This fact and availability of modern ink input devices have caused more attention has been given to the on-line handwriting recognition. However, the off-line recognition is still important, especially in scenarios where the strokes could not be captured, e.g. in recognition of historical documents.

This work deals with sketched diagrams, flowcharts specifically. Handwriting recognition in this domain is usually associated with new tools for natural and convenient design based on sketching using an ink input device. It is an alternative to classical drag-and-drop designers like Microsoft Visio. Therefore, the research is primarily focused on on-line diagram recognition. However, we see interesting applications of an off-line recognizer in scenarios like digitization of lecture notes or blackboard screenshots, as many users still prefer conventional media over ink input devices.

We have developed an on-line flowchart recognizer achieving the state-of-the-art performance [**?**] (improvements: [**?**], [**?**]). Our goal is to extend the recognizer to make it capable of recognizing flowcharts captured in static images. The reconstruction of strokes from images is a way how to use our existing recognizer for off-line recognition. Recovery of the dynamic information from static images is an active field of study. Research in cognitive psychology also suggests that human perception of dynamic information assists the recognition [2]. Interesting results have been achieved in the domain of cursive handwriting, where researchers often exploit natural properties of the handwriting as continuity or curvature smoothness.

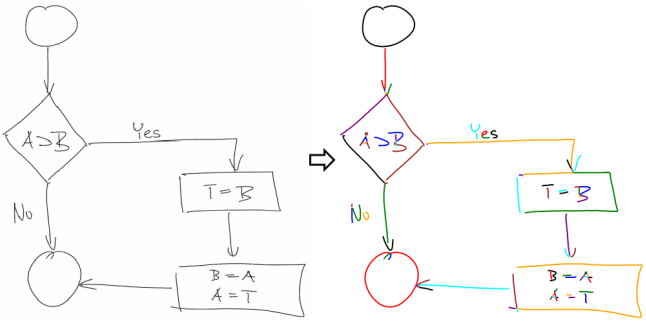


Figure 1: Strokes reconstructed from a flowchart image. Colors alter in the stroke sequence for better visualization.

The first contribution of this work is an off-line benchmark database of sketched flowcharts. We created it as an extension of our existing on-line database. We defined a mapping between the dynamic strokes and the static images. This database can be used for research in on-line and off-line recognition as well as in strokes reconstruction. Similar dual benchmark databases exist for cursive handwriting, e.g. IRONOFF containing individual words [3]. To the best of our knowledge, there is no dual benchmark database focused on graphical content like sketched diagrams.

The second contribution is the off-line flowchart recognizer created as a combination of the strokes reconstruction algorithm and the on-line flowchart recognizer. Our proposed strokes reconstruction algorithm builds on basic concepts from literature. Although it is simple and it serves as a baseline, the recognition results of the diagram recognizer are

very promising. Figure 1 shows an example of reconstructed strokes from a flowchart image. The reconstructed strokes lose their natural ordering. Therefore, the on-line recognizer needs an adaptation, because some steps of its recognition pipeline rely on the strokes ordering.

The rest of the paper is organized as follows. Work related to strokes reconstruction and off-line diagram recognition is summarized in Section II. Section III describes how we created the off-line benchmark database from the on-line one. Section IV introduces the used algorithm for strokes reconstruction from images. Section V shows how we adapted our on-line recognizer to work with reconstructed strokes where the natural writing order is lost. The experimental evaluation is presented in Section VI, a conclusion in Section VII.

II. Related Work

A survey on techniques for trajectory recovery from static handwriting was created by Nguyen and Blumenstein [2]. It confirms that on-line handwriting recognition systems are usually better than their off-line counterparts. The success of on-line systems encourage the recovery of dynamic information. Strokes reconstruction is done in three main steps: preprocessing, local examination and global examination.

The preprocessing usually includes gray-scale conversion, binarization, noise removal, and contour smoothing. Many algorithms work with a skeleton image [4], [5], where the strokes are one pixel wide. In such case, a thinning algorithm [6], [7] needs to be used. Although thinning simplifies the task, some information is lost during the skeletonization and thus there are methods working with stroke contours [8], [9]. Even the binarization may deteriorate valuable clues such as intensity consistency, continuation, and feathering, which could be utilized [10], [11].

The local examination includes detection and pairing of endpoints, detection of retraced writing and hidden loops, detection and analysis of ambiguous zones. The most challenging task is the ambiguous zones analysis, where the incoming and outgoing segments branching from ambiguous zones are matched. This operation is based on evaluation of continuity and smoothness of the resulting curve [8], [5].

The global examination is needed to resolve some remaining ambiguities and to put reconstructed strokes into the correct writing order. A graph search approach is employed, where endpoints, junctions, and touching points are represented by vertices and curves are represented by edges. The correct stroke sequence is determined by seeking the most appropriate path. The path can be found optimally by finding the Hamilton path (NP-hard problem), or sub-optimally using heuristics [4]. Criteria used to determine the optimal path are based on smoothness of the resulting curve or the minimal transition cost (handwriting energy) [12]. However, it is unlikely to use such criteria successfully in the domain of flowcharts where the sequence of strokes does not exhibit smoothness and individual users have specific writing orders. Sharp direction changes and compositions of crossing line segments occur often. Therefore, we do not expect recovering the correct writing sequence. We show that it is not crucial for the diagram recognition domain.

The problem of verifying the reconstructed strokes and comparing various algorithms was addressed by Nel et al., [13]. Recovered stroke sequences often do not match their dynamic counterparts exactly, but algorithms often recover strokes consistently. The quality of the strokes reconstructed is often verified manually by human observers. Nel et al. proposed a technique, where hidden Markov models are derived from the ground-truth strokes and the estimated strokes are then matched to the derived models. However, the ultimate goal is to apply on-line recognition techniques to off-line data and thus it makes sense to express the success of the stroke reconstruction indirectly through the precision of applied on-line recognizers. We adopt this technique.

The problem of the off-line flowchart recognition via strokes reconstruction was tackled by Wu et al. [14]. However, they assumed strokes to be already reconstructed from the image. They used the original strokes from the on-line database [15] and ignored their ordering. This makes the task significantly easier, since the strokes reconstruction is the most difficult part. Still, symbol segmentation in unordered strokes is an interesting problem, because this ordering is essential for on-line recognition techniques to efficiently reduce the search space when searching for symbol candidates. They introduced the idea of *shapeness* estimation. It basically exploits the property of the symbols in flowcharts – they form closed loops. The main idea is to use a simple and very fast classifier to reject candidates, which do not look like a closed loop. The classifier is based on compact features efficiently extracted from a candidate normalized into a very small image.

A pure off-line recognizer not performing strokes reconstruction was proposed by Notowidigdo and Miller [16].

III. Benchmark Database

We created two publicly available off-line databases[1] extending the existing on-line database[2]. The first database (abbreviated D_α) consists of images with noise-free 1-pixel wide rasterization of strokes (skeleton images). The second database (abbreviated D_β) contains scans of a printed thicker stroke visualization containing real noise. Having these two databases allows us to investigate the impact of the skeletonization algorithm on the recognition results.

D_α was created as follows. The page A4 has a printable surface $247 \times 170\,\mathrm{mm}$, landscape. We used a canvas $2\,918 \times 2\,008$ pixels to obtain $300\,\mathrm{dpi}$ when the image is printed. We left 10 pixels on each side for margins. We scaled the strokes to fit the canvas and centered them. The strokes were

[1] http://cmp.felk.cvut.cz/~breslmar/flowcharts_offline/

[2] http://cmp.felk.cvut.cz/~breslmar/flowcharts/

rasterized in the image using the Bresenham's algorithm. Additionally, we created markers 5 pixels from the top-left and bottom-right corner of the bounding box of the strokes. The markers might be used to find a mapping of the original strokes onto the image.

D_β was created as follows. We used WeInspire[3] sketching application to visualize the strokes and saved them into PDF files. The canvas in the application is $1\,596 \times 922$ pixels. We scaled and centered the strokes to fit inside as above. We drew a bounding rectangle 15 pixels far from the strokes bounding box. We printed and then scanned (All-in-One KONICA MINOLTA bizhub C3110) the diagrams to obtain images with natural noise, which do not significantly differ from scans of diagrams drawn using a standard pen and paper. We used the bounding rectangle to estimate the global transformation mapping the original strokes onto the image.

For each InkML file with the on-line data, there is a PNG file with the image and XML file with an off-line annotation. The off-line annotation contains positions of the marker points in the image, distance of the marker points from the bounding box in the original image (before scanning), and parameters (scale, rotation, and rotation) of the transformation, which maps the original strokes in the InkML file onto the strokes in the image. The annotation of symbols and relations between them follows. Each symbol has a bounding box defined and a symbol class assigned. Text blocks have their meaning assigned and arrows have a bounding box of their head and connection points defined. Relations are defined the same way as in the on-line database.

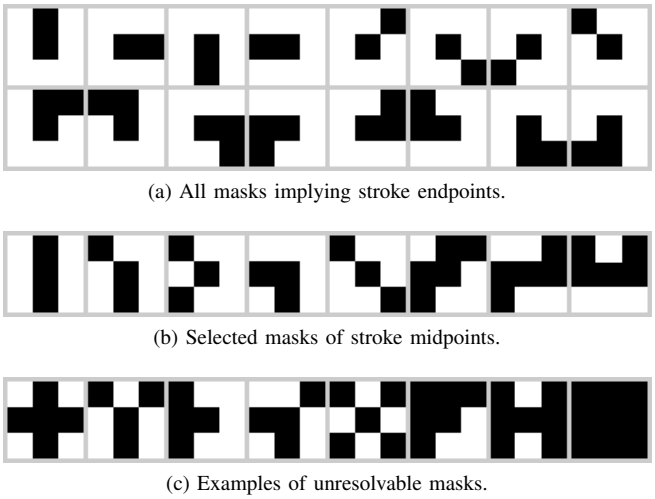
IV. Strokes Reconstruction

The proposed algorithm builds on common techniques described in the related work and serves as a baseline. It works with the skeleton image as the input. We use the following thinning algorithm when recognizing diagrams from D_β: the image is smoothed using a Gaussian blur (3×3 window, $\sigma = 3.0$), binarized using a simple thresholding (threshold 0.6), and thinned using the Zhang-Suen algorithm [6].

The strokes reconstruction is performed separately for each connected component. Each pixel of the skeleton is labeled as an ambiguous or unambiguous based on its 3×3 neighborhood. The idea behind this is that we can clearly decide (up to the direction) what is the trajectory of the stroke within a 3×3 window if it fits one of the 80 predefined patterns (out of 512 possible contents). 16 of these patterns correspond to stroke endpoints. See Figure 2 for examples.

The ambiguous pixels usually form the whole ambiguous zones, which occur where the strokes are touching, crossing, or one stroke is retraced. We group touching ambiguous pixels together to form these zones. Chains of unambiguous points separated by ambiguous zones or empty space are traced to create strokes. If a stroke is touching the ambiguous zone, it is attached to it to be resolved later. The perfect strokes reconstruction would requite to correctly resolve all the ambiguous zones.

[3]https://we-inspire.com/



(a) All masks implying stroke endpoints.

(b) Selected masks of stroke midpoints.

(c) Examples of unresolvable masks.

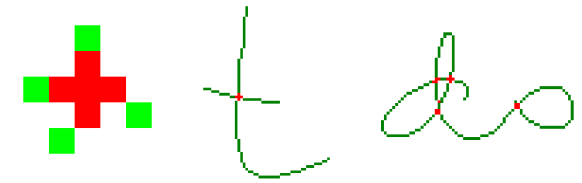
Figure 2: Illustration of the neighborhood masks used to identify unambiguous pixels: a) endpoints, b) midpoints. Examples of possible ambiguous masks are shown in c).

We analyzed all the ambiguous zones from the training dataset of D_α. We identified over $9\,\mathrm{k}$ unique ambiguous zones. We annotated 626 of the most frequent ones (the most and the least frequent zones, which we annotated, appeared $1\,761$ and 6 times, respectively), which can be resolved the same way in all of their occurrences. The annotation contains information which strokes incident with the zone are linked together and what are the trajectories of these links. These annotations are used to resolve some portion of the ambiguous zones and thus to iteratively merge pairs of strokes belonging together. Examples of resolvable and unresolvable annotated ambiguous zones are shown in Figure 3. Unresolved ambiguous zones cause that some original strokes fall apart. Extremely short strokes, partially caused by hooks at stroke endings, are discarded. The additional effect is that retraced strokes are simplified. However, these situations occur more often at text strokes, which are not crucial for the recognition of the diagram structure.

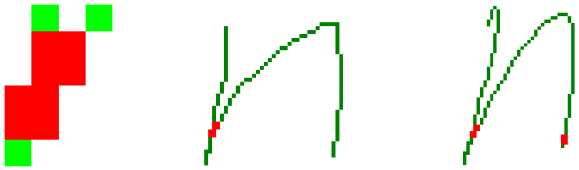
V. Recognizer Adaptation

The recognition pipeline of our on-line diagram recognizer consists of the following steps: 1) text / non-text separation, 2) segmentation of uniform symbols, 3) classification of segmented symbol candidates, 4) arrows detection, 5) structural analysis, 6) text blocks recognition. For more details about the individual steps, see our paper [**?**].

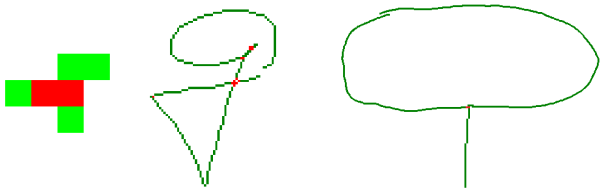
The way how users draw diagrams defines a natural ordering of strokes. It usually holds that individual text blocks or symbols are formed from consecutive strokes. This property is used in two steps of the recognition pipeline:



(a) Ambiguous zone caused by crossing strokes.



(b) Ambiguous zone caused by a retraced stroke.



(c) Ambiguous zone, which is sometimes caused by crossing strokes and sometimes by a retraced stroke.

Figure 3: Examples of ambiguous zones in the left column and their occurrences in the next two columns: a) and b) are resolvable, c) is not resolvable – a vertical link merging the bottom and the top strokes is required in the first example, while a horizontal link is required in the second example.

text / non-text stroke classifier and symbol segmentation. The text / non-text classifier solves the problem as a sequence labeling and thus learns the natural stroke ordering. The context of consecutive strokes provides an important piece of information. The symbol segmentation is based on stroke clustering using a trained distance function which consists of several weighted features including the stroke ordering. However, the reconstructed strokes lack the information about this natural ordering. It cannot be deduced without understanding the diagram. The other parts of the recognition pipeline do not depend on the stroke ordering thus the adaptation of the recognizer consists of modifying the two mentioned steps. The recognizer works with an empirical distance threshold $\mathrm{distThresh}$ derived from the data (details in [**?**]). We say two points are close if the Euclidean distance between them is smaller than $\mathrm{distThresh}$.

A. *Text / non-text separation*

Our on-line diagram recognizer uses the text / non-text classifier proposed by Van Phan and Nakagawa [17]. It combines unary and binary features. The unary features are used to classify individual strokes into the classes text and non-text. The binary features capture relationships between adjacent strokes in the writing order. They are used to classify transitions between strokes into three classes: text–text, text–non-text, non-text–non-text. Since it can be seen as a sequence labelling task, Bidirectional Long Short Term Memory (BLSTM) Recurrent Neural Network (RNN) classifiers are used to capture the global context. The probabilistic outputs from the two BLSTM neural networks are combined together to obtain the final labeling probability.

Since the stroke ordering is lost, we need to change the neighborhood system. Instead of considering relationships between adjacent strokes in the writing order, we consider relationships between strokes spatially close to each other. Contrary to the former system, a stroke might have an arbitrary number of neighbors $n = 1, 2, \ldots, m-1$, where m is the number of reconstructed strokes. We apply a restriction that there are at most three closest strokes considered to be relevant neighbors. The unary and binary features extracted from individual strokes and pairs of neighboring strokes remain the same with one exception – features based on temporal information are omitted. Since the classification cannot be seen as the sequence labeling anymore, we use SVM classifiers. The final potentials of strokes to be in the class text and non-text are obtained by combining probabilistic outputs of individual SVMs evaluating the unary and binary features. As Van Phan and Nakagawa suggested, this combination is done using the sum rule as follows:

$$Po(T|s_i) = P(T|s_i) + \sum\nolimits_{j \in N} [P(T|s_i) \cdot P(TT|s_i, s_j) \cdot \\ \cdot P(T|s_j) + P(T|s_i) \cdot P(TN|s_i, s_j) \cdot P(N|s_j)],$$

$$Po(N|s_i) = P(N|s_i) + \sum\nolimits_{j \in N} [P(N|s_i) \cdot P(NN|s_i, s_j) \cdot \\ \cdot P(N|s_j) + P(N|s_i) \cdot P(TN|s_i, s_j) \cdot P(T|s_j)],$$

where N is the set of neighbors of the stroke s_i; $P(T|s_i)$ and $P(N|s_i)$ are the probabilities of s_i to be in the class text and non-text, respectively; $P(TT|s_i, s_j)$, $P(TN|s_i, s_j)$, and $P(NN|s_i, s_j)$ are the probabilities of the pair of neighboring strokes s_i, s_j to be in the class text–text, text–non-text, and non-text–non-text, respectively. Note that we do not differentiate between transitions text–non-text and non-text–text. Although the global context captured by the BLSTM RNN is partially lost, the information given by the neighboring system still allows to achieve satisfactory results. As we explain in [**?**], we want to achieve the minimal possible error rate for the class non-text and thus remove only strokes where the classifier is almost certain about them to be text. Therefore, we biased the result of the classifier. We increased the potential $Po(N|s_i) = Po(N|s_i) + 0.9; \forall i = 1, \ldots, m$ and achieved the precision in the classes text / non-text 78.67 % / 92.77 % in D_α and 82.61 % / 92.86 % in D_β.

B. *Uniform Symbol Segmentation*

Similarly to Wu et al. [14], we decided to exploit the property of flowchart symbols that they always comprise of strokes forming a closed loop. However, we do not assume

any particular shape of the formed loop. We only check its convexity and distances between endpoints of consecutive strokes forming the loop. These values are used to reject improper loops and to compute a confidence of promising loops. The loop detection works iteratively. First, we create initial loops of single strokes with the first and last points defined. In each further iteration, we search for strokes, which could be attached to unfinished loops. These strokes cannot violate the endpoints proximity, the loop convexity, and the maximal loop size: 1) One of the endpoints of the stroke to be added has to be close to the last point of the unfinished loop. If the other endpoint is close to the first loop point, the loop is closed and the average distance between consecutive strokes in the loop d_1 is computed. 2) The violation of convexity is measured as the maximal distance of any point from the convex hull. If this distance d_2 is bigger than the $\mathrm{distThresh}$, the loop is immediately discarded. 3) Each unfinished loop is discarded if it contains more than five strokes. The restrictions 1)-3) reduce the search space greatly. The final confidence of the loop is

$$\mathrm{conf}_{\mathrm{loop}} = 1.0 - \frac{d_1 + d_2}{2 \cdot \mathrm{distThresh}} \qquad (1)$$

C. Rest of the Recognition Pipeline

The reconstructed strokes differ from the original ones and thus we retrained all classifiers in the recognizer using the annotation and reconstructed strokes from D_α. The recognition continues according to the unchanged remainder of the recognition pipeline. The detected loops are classified by the symbol classifier. Arrows are detected as connectors between pairs of the found symbols. Symbols with arrows form a set of symbol candidates. The structural analyser chooses a subset of symbol candidates, which forms a diagram the best. Finally, the recognized diagram structure helps to form unused strokes into text blocks.

VI. Experiments

Our algorithm for strokes reconstruction does not provide the perfect reconstruction and thus the reconstructed strokes do not perfectly match the original ones. Usually, the reconstructed strokes are sub-strokes of the original strokes separated by the unresolvable ambiguous zones. This shattering of the strokes does not necessarily prohibit correct recognition. However, it increases the processing time. Instead of direct measurement of a reconstruction quality, we rather compare the recognition results of the original recognizer on the on-line data with the adapted recognizer on the off-line data using the strokes reconstruction. The quality of the strokes reconstruction affects indirectly the performance of the recognizer.

We use two common metrics to make the comparison with the annotated ground truth: First, correct stroke labeling rate (SL). Second, correct rate of symbol segmentation and recognition (SR). Each annotated symbol is considered to be correctly segmented and recognized if there is a match among detected symbols. Two symbols match if the overlap of their bounding boxes is at least $80\,\%$ and the labels match. Arrows are additionally required to be connected to the same matched symbol. Results are summarized in Table I. We also measure the number of unrecognized symbols in each diagram. $44.9\,\%$ of diagrams in D_β were recognized correctly, see the histogram in Figure 4.

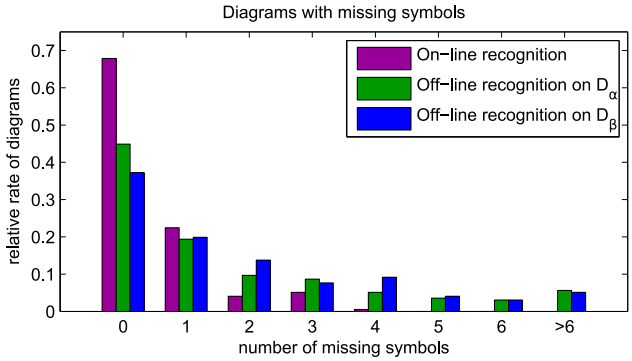


Figure 4: Error rates by the number of missing symbols.

Class	SL [%]			SR [%]		
	On.	D_α	D_β	On.	D_α	D_β
Arrow	93.8	86.2	85.1	94.3	85.1	84.3
Connection	88.4	68.7	61.0	89.3	84.8	86.6
Data	96.1	83.1	79.7	94.7	95.8	94.4
Decision	90.3	83.5	83.2	95.1	96.9	96.9
Process	98.4	88.7	88.6	98.4	99.2	98.8
Terminator	99.7	74.0	71.9	99.6	94.0	93.6
Text	99.6	99.5	99.5	98.7	95.1	93.7
Total	98.4	95.9	95.0	98.7	92.33	91.3

Table I: Results achieved for the on-line data, D_α, and D_β.

We measured the running time of the adapted recognizer on the off-line data. The average running time was $6.4\,\mathrm{s}$ and $11.0\,\mathrm{s}$ for D_α and D_β, respectively. The detailed measurements are in Table II. The most time consuming steps are strokes reconstruction and loops detection. However, we did not attempt to parallelize or optimize these steps, which is possible. Moreover, we argue that the off-line recognition does not require real-time performance.

To make a comparison with the recognizer by Wu et al. [14], we tested our adapted recognizer on unordered strokes from the FC database by Awal et al. [15]. We achieved the precision of $95.6\,\%$ and $84.2\,\%$ in correct strokes labeling and symbol recognition, respectively. The average running time was $1.24\,\mathrm{s}$. The result is deteriorated in comparison with our original recognizer relying on the stroke ordering ($96.3\,\%$, $85.4\,\%$, $0.78\,\mathrm{s}$, respectively). However, it is slightly better than the result achieved by Wu et al., which was $94.9\,\%$ and $83.2\,\%$ for stroke labeling and symbol recognition, respectively.

Step	D_α [ms]			D_β [ms]		
	min	max	avg	min	max	avg
1)	462	10 360	2 362	1 696	19 781	6 354
2)	16	190	73	16	215	76
3)	155	52 089	2 437	143	15 801	2 915
4)	508	3 544	1 549	493	4 743	1 624
Total	1 440	64 922	**6 422**	2 623	37 972	**10 970**

Table II: Running time in milliseconds consumed by the individual steps: 1) strokes reconstruction, 2) text / non-text separation, 3) loops detection, 4) rest of the pipeline.

VII. Conclusion

We have demonstrated that the effort put into the on-line recognition, whose research receives more attention nowadays, can be efficiently reused for the off-line recognition. It turned out that a simple algorithm for strokes reconstruction is sufficient to obtain a satisfactory results for sketched flowcharts. In addition, only small modifications of the existing on-line recognizer were required. We encourage the reader to experiment with our demo application[4].

We have also contributed a new database of off-line data, being a counterpart to existing on-line data. The experiments performed on it have established a solid baseline. We believe the database can be used by researches to evaluate off-line flowchart recognizers as well as techniques for strokes reconstruction from images.

References

[1] M. Bresler, D. Průša, and V. Hlaváč, "Online recognition of sketched arrow-connected diagrams," *International Journal on Document Analysis and Recognition (IJDAR)*, pp. 1–15, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10032-016-0269-z

[2] V. Nguyen and M. Blumenstein, "Techniques for static handwriting trajectory recovery: A survey," in *DAS '10: 9th IAPR International Workshop on Document Analysis Systems*. New York, NY, USA: ACM, 2010, pp. 463–470.

[3] C. Viard-Gaudin, P. M. Lallican, S. Knerr, and P. Binter, "The IRESTE on/off (IRONOFF) dual handwriting database," in *ICDAR '99: Fifth International Conference on Document Analysis and Recognition*, Sep 1999, pp. 455–458.

[4] Y. Kato and M. Yasuhara, "Recovery of drawing order from single-stroke handwriting images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 9, pp. 938–949, Sep. 2000.

[5] Y. Qiao, M. Nishiara, and M. Yasuhara, "A framework toward restoration of writing order from single-stroked handwriting image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1724–1737, Nov 2006.

[6] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984.

[4] http://cmp.felk.cvut.cz/~breslmar/flowcharts_recognizer_offline/

[7] H. Chatbri and K. Kameyama, "Towards making thinning algorithms robust against noise in sketch images," in *ICPR '12: 21st International Conference on Pattern Recognition*, Nov 2012, pp. 3030–3033.

[8] R. Plamondon and C. M. Privitera, "The segmentation of cursive handwriting: an approach based on off-line recovery of the motor-temporal information," *IEEE Transactions on Image Processing*, vol. 8, no. 1, pp. 80–91, Jan 1999.

[9] T. Steinherz, D. Doermann, E. Rivlin, and N. Intrator, "Offline loop investigation for handwriting analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 193–209, Feb 2009.

[10] B. Kovari, "Time-efficient stroke extraction method for handwritten signatures," in *ACS '07: 7th WSEAS International Conference on Applied Computer Science*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society, 2007, pp. 157–161.

[11] V. Pervouchine and G. Leedham, *Document Analysis Systems VII: 7th International Workshop, DAS 2006, Nelson, New Zealand, February 13-15, 2006. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ch. Extraction and Analysis of Document Examiner Features from Vector Skeletons of Grapheme 'th', pp. 196–207.

[12] K. T. Tang and H. Leung, "Reconstructing strokes and writing sequences from chinese character images," in *International Conference on Machine Learning and Cybernetics*, vol. 1, Aug 2007, pp. 160–165.

[13] E.-M. Nel, J. A. du Preez, and B. M. Herbst, "Verification of dynamic curves extracted from static handwritten scripts," *Pattern Recogn.*, vol. 41, no. 12, pp. 3773–3785, Dec. 2008.

[14] J. Wu, C. Wang, L. Zhang, and Y. Rui, "Offline sketch parsing via shapeness estimation," in *IJCAI '15: 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 1200–1206.

[15] A.-M. Awal, G. Feng, H. Mouchere, and C. Viard-Gaudin, "First experiments on a new online handwritten flowchart database," in *DRR'11*, 2011, pp. 1–10.

[16] M. Notowidigdo and R. C. Miller, "Off-line sketch interpretation," in *AAAI Fall Symposium Series*. AAAI Press, 2004.

[17] T. V. Phan and M. Nakagawa, "Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents," *Pattern Recognition*, vol. 51, pp. 112 – 124, 2016.