18^{th} *Computer Vision Winter Workshop*
Walter G. Kropatsch, Fuensanta Torres, Geetha Ramachandran (eds.)
Hernstein, Austria, February 4-6, 2013

Simultaneous Segmentation and Recognition of Graphical Symbols using a Composite Descriptor

Martin Bresler, Daniel Průša, Václav Hlaváč
Czech Technical University in Prague, Faculty of Electrical Engineering
Department of Cybernetics, Center for Machine Perception
166 27, Praha 6, Technická 2, Czech Republic
{breslmar, prusapa1, hlavac}@cmp.felk.cvut.cz

Abstract. *This work deals with recognition of hand-drawn graphical symbols in diagrams. We present two contributions. First, we designed a new composite descriptor expressing overall appearance of symbols. We achieved rather favorable accuracy in classification of segmented symbols on benchmark databases, which is 98.93% for a database of flow charts, 98.33% for a database of crisis management icons, and 92.94% for a database of digits. Second, we used the descriptor in the task of simultaneous segmentation and recognition of graphical symbols. Our method creates symbol candidates by grouping spatially close strokes. Symbol candidates are classified by a multiclass SVM classifier learned on a dataset with negative examples. Thus, some portion of the candidates is filtered out. The joint segmentation and classification was tested on diagrams from the flowchart database. We were able to find 91.85% of symbols while generating 8.8 times more symbol candidates than is the number of true symbols per diagram in average.*

1. Introduction

The proliferation of tablets or tablet PCs implies the demand for algorithm allowing interface by handwriting or hand-drawing. The attention of researches is recently given on graphical representation of human thoughts, e.g. diagrams. One of the most common diagram for various branches is a flowchart. It is used to describe a general algorithm or a process. The flowchart is composed of boxes connected by arrows and text, which can be inside the boxes or can label the arrows. See the example of flowchart in Figure 1. To recognize a flowchart fully, we have to recognize all symbols correctly, find relations between

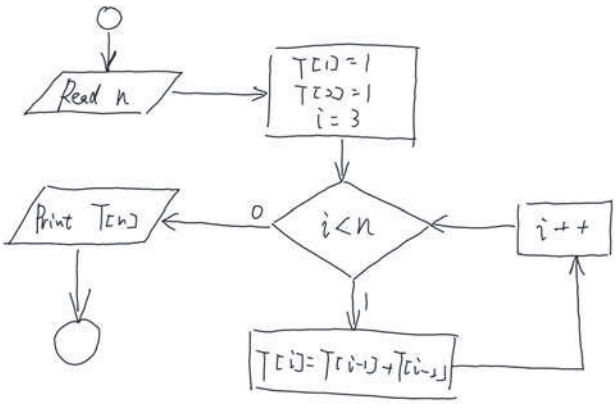


Figure 1. Example of a flowchart from the FC database.

them, and also recognize a text.

Each diagram recognizer must perform following six stages [4]: (1) early processing - noise reduction, de-skewing, etc., (2) segmentation of strokes into isolated symbols, (3) symbol recognition, (4) identification of spatial relationships among symbol, (5) identification of logical relationships among symbols, (6) semantic interpretation. This paper describes our approach to perform steps (2) and (3) of the diagram recognition pipeline. We allow multiple candidates for the symbols to be created. The segmentation phase thus does not make a final decision. This is left at the next stages performing structural analysis. We follow the structural construction paradigm proposed by Schlesinger and Hlaváč [13]. We chose flowcharts as diagrams of our interest and we used FC database [1] for training and verification. It contains 327 diagrams drawn by 35 users and there are 4 780 symbols. The database is divided into a training dataset (200 diagrams, 2 919 symbols) and a test dataset (127 diagrams, 1 861 symbols). The diagrams are stored in inkml file format, where are in-

dividual strokes and symbols defined. We recognize just graphical symbols and text must be processed in different way. Therefore, the text is filtered out from the diagrams. In practice, this can be done using various text / non-text stroke classifiers. The state-of-the-art methods work with a high accuracy [3], [17], [7], [11].

Symbols segmentation was researched by many people. For example, Kara and Stahovich [8] presented an approach where arrows are detected first and they are stated immediately as a ground truth. Then the rest represents separated symbols. Peterson et al. [12] came with two-step stroke grouping based on a single stroke classification followed by clustering of strokes within the classes. Although the results are promising, there is still much work to do to solve the problem robustly. There exist algorithms for graphical symbols recognition, which classify already segmented symbols. These algorithms are often based on HMM or MRF [2], [18]. Some algorithms work with more complex graphs representations [9]. In some cases, it is beneficial to combine more approaches together to achieve higher accuracy [14]. The results vary with respect to the difficulty of the problem, i.e. how complex the symbols are, what is the quality of databases etc. We present an approach, in which the segmentation and classification are performed simultaneously. The segmentation is not final and we rather generate symbol candidates for the next stage of the recognition pipeline. Therefore, we call it pre-segmentation. The approach is simultaneous in the sense that the pre-segmentation is a result of classification of selected groups of strokes. This is the main advantage in comparison with the state-of-the-art methods. They do hard decisions in symbol segmentation step or rather focus on recognition of already segmented symbols.

The rest of the paper is organized as follows. The method of segmentation by classification is described in Section 2. Our symbol descriptor is defined in Section 3. An evaluation of experiments on the FC database and experiments on additional databases are given in Section 4 to show the generality of the descriptor. Finally, our conclusions are presented in Section 5.

2. Pre-segmentation by Classification

This section describes our approach how to group strokes of a flowchart into isolated symbols and how to classify them. The main idea is that we create symbol candidates by grouping single strokes together followed by classification of all the candidates. We assume there is no stroke which is common to two or more symbols. This assumption holds in most of the cases. Our goal is to generate as few candidates as possible and miss as few true symbols as possible at the same time. The generated symbol candidates are supposed to be filtered in the next stages of the pipeline using information about relations between them.

2.1. Strokes Grouping

First, we take all single strokes as a symbol candidates of size 1. Then, we create iteratively new symbol candidates of size n by adding a single, spatially close, stroke to symbol candidate of size $n-1$. It can be seen in the histogram of true symbol sizes in the FC database (see Figure 2), that it is sufficient to create symbol candidates of maximal size 5. Two strokes (or a stroke and a group of strokes) are spatially close if the distance between two closest points is below a threshold, which is defined as $distThresh = k \cdot D_{med}$, where D_{med} is a median of lengths of diagonals of bounding boxes of all single strokes in a diagram. The usage of the D_{med} makes the $distThresh$ independent of the overall size of the diagram, which differs due to the different writer's conventions or different resolution of used devices. We chose the value of k to be 0.35 empirically (see Figure 3).

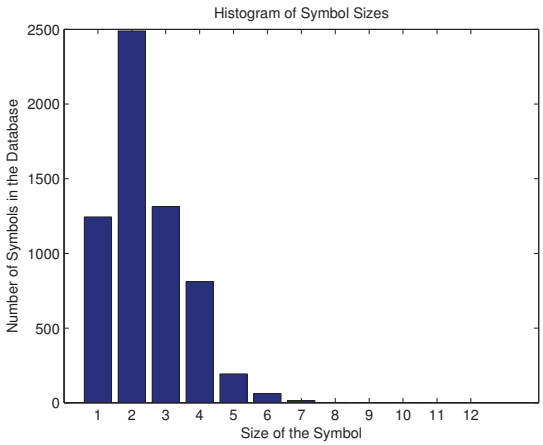
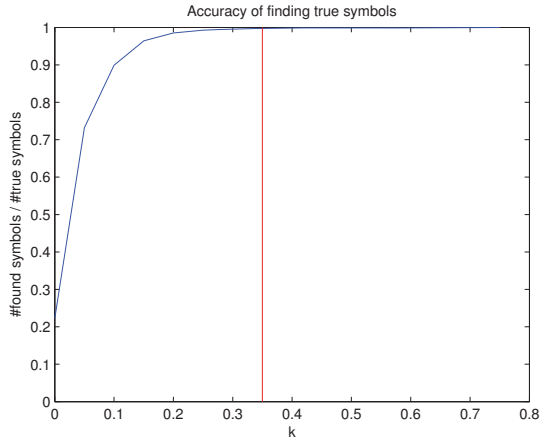


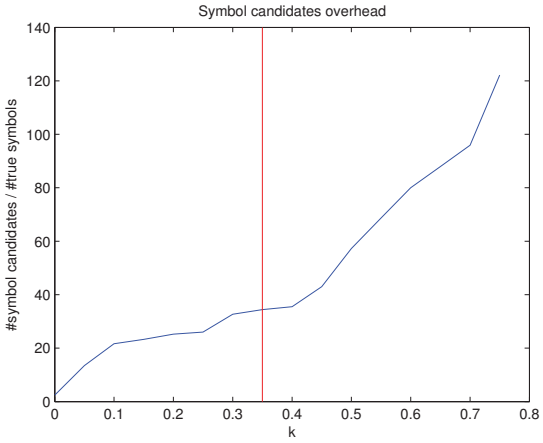
Figure 2. Histogram of Symbol Sizes.

2.2. Classification

The classification of symbol candidates is based on a descriptor which is described in Section 3. We use a multiclass classifier implemented as an instance



(a)



(b)

Figure 3. Result of the experiments on the whole FC database to obtain optimal value of the k coefficient. (a) shows the growth of the accuracy with growing k and (b) shows the growth of the number of symbol candidates. The choice of optimal k is a tradeoff.

of a structured output SVM learned by BMRM algorithm [16]. A logistic regression is fitted on the classifier response to obtain a posterior probability that a symbol candidate belongs to the class. We used Statistical Pattern Recognition Toolbox for Matlab (STPRtool) [5] for this.

We have seven basic classes in our training data: *arrow*, *connection*, *data*, *decision*, *process*, *terminator*, and *no_match*. Examples of objects from these classes are shown in Figure 4. The last class represents symbol candidates with no meaning. It is very important to prevent false positive detections. The training data is extracted from the training diagrams of the FC database. We find all symbol candidates of maximal size 3 in a way described above. The labels are taken from the annotated database for symbol candidates which represent true symbols. The rest of the candidates has no meaning and thus gets the label *no_match*. In total, we have 32 064 symbols in the training dataset, where are 1 474 examples of class *arrow*, 143 of *connection*, 337 of *data*, 248 of *decision*, 477 of *process*, 240 of *terminator*, and 29 145 *no_match*.

Symbols of the same class may look very differently yielding different descriptors. It is the most common for classes *arrow* and *no_match* (see Figure 4). Therefore, we cluster symbols of these classes into several clusters according to their descriptors. We use 10 clusters for arrows, 30 clusters for the class *no_match* and two clusters for the rest. We define a loss function which gives higher penalty when a symbol is classified as *no_match*. Specific values of the loss function are shown in Table 1. Classification into a different cluster of the same class does not count as an error. Thus, the table defining the loss function is simplified.

	arr.	conn.	data	dec.	proc.	term.	no_m.
arr.	0	2	2	2	2	2	1
conn.	2	0	2	2	2	2	1
data	2	2	0	2	2	2	1
dec.	2	2	2	0	2	2	1
proc.	2	2	2	2	0	2	1
term.	2	2	2	2	2	0	1
no_m.	100	100	100	100	100	100	0

Table 1. Definition of the loss function. Each column represents a true class.

We used the cross-validation with 5 folders to obtain the optimal value of the regularization constant for the training. The data was divided into five folders w.r.t. their classes and four folders was used for training and one for testing. We chose the constant to be 10^{-7} according to the cross-validation test error. Then we used the constant and learned the classifier on the whole train dataset. Results of the classifier on test diagrams of the FC database are shown in Section 4.

3. Descriptor

As we have shown in Section 1, there are many ways how to describe a hand-drawn graphical symbol. Flowcharts are composed of very simple and geometrically describable symbols such as rectangles, circles, etc. See Figure 1. Those symbols, of course, may change the size arbitrarily. Moreover, they can be drawn with a different number of strokes in arbitrary order. Therefore, a descriptor which requires an

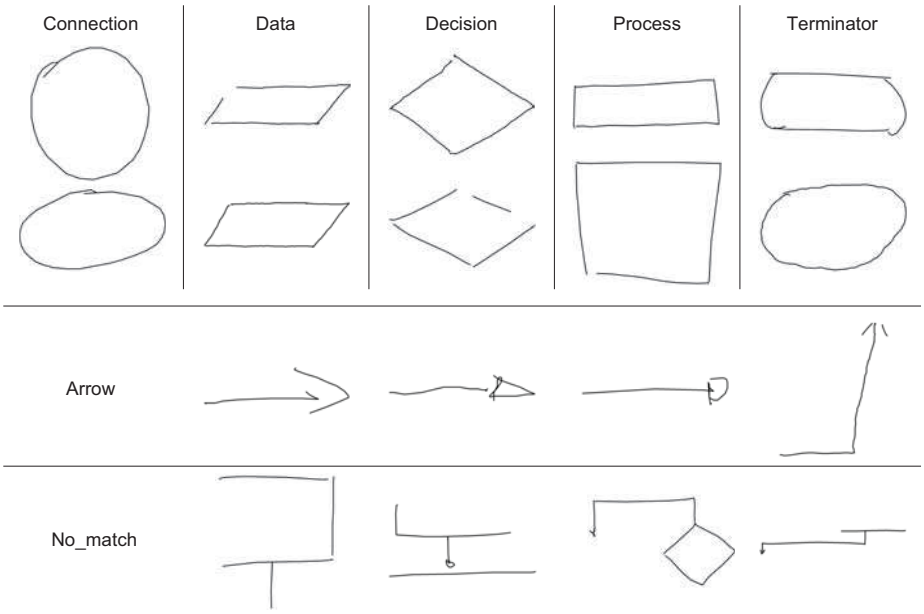


Figure 4. Examples of symbols from each of the classes taken from the database.

exact order of strokes would imply unnatural requirements on the user. For the purpose of recognizing graphical symbols contained in a flowchart, we proposed a new descriptor. Although, it is designed for flowcharts it can be used to describe arbitrary graphical symbol.

3.1. Descriptor Components

Our descriptor is composed of three components. The first one is the normalized histogram of distances between points. The second one is the normalized histogram of angles given by three points, and the last component is the histogram of small substrokes (compositions). Each of those components does not have power to fully describe the overall appearance of a given symbol. However, their combination showed to be discriminative enough. The dimension of the whole descriptor is 90 which is a concatenation of its three components with dimension 32, 16, and 42, respectively.

As we mentioned before, a symbol may be composed of many strokes. Therefore, we take points of all strokes and ignore gaps between endpoints. Let us denote the sequence of points representing a symbol $P = \{p_1, p_2, \ldots, p_n\}$.

3.1.1 Histogram of Distances Component

The distance between each possible pair of points on x-axis (Δx) and y-axis (Δy) is computed and these values are assigned to corresponding histograms. This descriptor component is composed of four histograms with 8 bins each, where two histograms are allocated for Δx and two histograms for Δy. The Algorithm 1 shows how to assign Δx to its two corresponding histograms. The approach is analogous for Δy. All histograms are finally normalized by the number of pairs of points $\binom{n}{2}$.

Data: P, Δx, $lowHist$, $highHist$
Result: Increment certain bin in either $lowHist$ or $highHist$
$lowThresh = \max_{i,j,i\neq j,j-i=1} |p_i.x - p_j.x|$;
$highThresh = BoundingBox(P).width$;
if $\Delta x < lowThresh$ **then**
  $binSize = lowThresh/8$;
  $bin = \Delta x / binSize$;
  $lowHist[bin] = lowHist[bin] + 1$;
else
  $binSize = highThresh/8$;
  $bin = \Delta x / binSize$;
  if $bin > 7$ **then**
    $bin = 7$;
  end
  $highHist[bin] = highHist[bin] + 1$;
end

Algorithm 1: Assignment of a Δx value to the certain histogram.

3.1.2 Histogram of Angles Component

The second component expresses the curvature of the symbol. Triplets of points p_i, p_j, p_k are taken such that $j - i > 1, k = \lfloor (i + j)/2 \rfloor$. For all these triplets we compute the angle

$$\alpha = \arccos \frac{\overrightarrow{p_j p_k} \cdot \overrightarrow{p_k p_i}}{\|\overrightarrow{p_j p_k}\| \cdot \|\overrightarrow{p_k p_i}\|}. \quad (1)$$

Values are mapped to 16-bin histogram the standard way. The size of the bins is $\frac{\pi}{16}$. The histogram is normalized by the number of angles $\binom{n-1}{3}$.

3.1.3 Histogram of Compositions Component

The last component is inspired by the work of Tabernik et al. [15], in which the histogram of compositions (HoC) was presented as a low-level image descriptor. We used the similar principle in building a low-level stroke descriptor. First, angles between vectors given by two consecutive points and x-axis are computed. These angles are components of the first layer. All pairs of consecutive components of the previous layer are combined to create new compositions of the next layer. The compositions carry only information about the angles. The first two layers are visualized in Figure 5. Empirically, we chose to quantize the first level components to six values in the first layer and to use two layers. Higher number of both, components of the first layer and number of layers, leads to the lower accuracy because the descriptor would be too specific and less tolerant to different styles of drawing. Therefore, this component of the descriptor has dimension 42 since it contains the 6-bin histogram of first layer components and the 36-bin histogram of second layer components. When the histograms are computed, each angle is linearly interpolated into two neighbouring bins. Therefore, each first layer component contributes to two bins of the corresponding histogram and each second layer composition contributes to four bins. Both histograms are normalized to sum into 1.

It might be computationally expensive to compute the descriptor since all possible pairs of points are considered in the first two components. In the situation, in which the symbols are more complex (consist of higher number of points), it is possible to skip some pairs. We were able to use only 50% of possible pairs (leads to double speedup) without loss of the accuracy in our experiments.

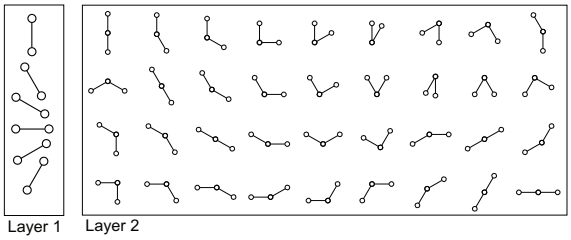


Figure 5. Visualization of compositions in two layers.

4. Experiments

We present our results achieved with the descriptor and the SVM classifier. First, the results of experiments with segmentation and classification of entire diagrams in FC database are presented. Later, we show results of experiments with the classification of segmented symbols from various databases, which demonstrate the universality of our descriptor.

We also measured how fast the computation of our descriptor is for the segmented symbols from the FC database. The symbols consist of 76.32 points in average and the computation time was in average 1.83 ms per symbol. The code is implemented in C#. We ran it on a standard tablet PC Lenovo X61 (Intel Core 2 Duo 1.6 GHz, 2 GB RAM) with 64-bit Windows 7 operating system.

4.1. Recognizing Diagrams

We tested our learned classifier on the testing diagrams of the FC database. The goal was to find as many true symbols as possible while keeping the number of symbol candidates as low as possible. The database contains 127 testing diagrams with 14.7 symbols in average. The strokes grouping algorithm is able to find 99.7% of the symbols in average while generating 368.9 symbol candidates in average. All symbols candidates were classified and three possibilities with the highest posterior probability were taken for each symbol candidate forming the set of classification results. Classification results with label *no_match* were removed. Results with posterior probability lower than 0.001 were removed as well. A possible post-processing step is to set a maximal number of classification results for each stroke and remove excessive ones with the lowest probability. The results can be found in the Table 2. Without removing excessive classification results, we can obtain accuracy 91.9% with 129.2 classification results in average. How many symbols from each class were

not recognized shows Table 3. The class arrow has the highest number of unrecognised symbols. It is because an arrow can have different shapes (different head, direction, etc.) and thus the recognition is more difficult. Achieved number of classification results is acceptable for the next step of the pipeline.

max. #candidates / stroke	#candidates	accuracy
5	48.31	86.20%
8	66.25	88.65%
10	75.54	89.45%
12	83.30	89.89%
15	92.97	90.40%
∞	129.17	91.85%

Table 2. Dependence of the average number of generated symbol candidates and the accuracy on the maximal number of symbol candidates per stroke.

arrow	connection	data	descision	process	terminator
72	0	14	18	24	32

Table 3. Numbers of unrecognised symbols from each class (out of all 1 861 symbols).

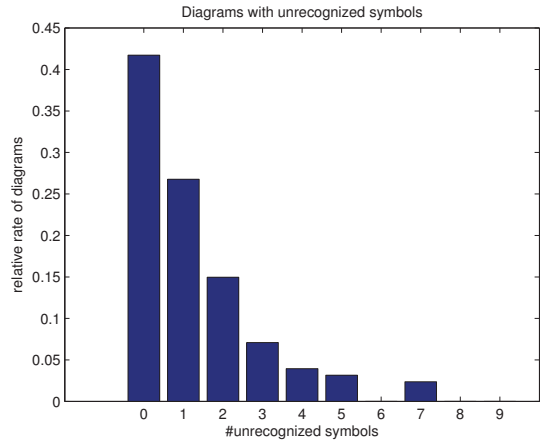


Figure 6. Histogram of numbers of unrecognised symbols in the diagrams.

Although the accuracy is not very high, it can be seen in Figure 6 that there are many diagrams (around 45%) where the accuracy 100% was reached. Those diagrams were drawn by users with nice drawing style, which is easily readable. On the other hand, diagrams with very low accuracy are sometimes very hardly readable even by human. In some cases we are facing bad annotation too. Therefore, we consider the results encouraging. Moreover, we expect that the recognition system will be providing a tool for a quick correction of misclassifications by the user. While the number of misclassifications is low (one or two) the system should be still effectively usable.

4.2. Classification of Segmented Symbols

We tested our descriptor on different databases of hand-drawn graphical symbols to get an idea how discriminative it is and for what kind of symbols it can be used. The symbols were segmented. In all cases, we used SVM classifier trained by BMRM on training dataset of the database and tested the classifier on the test dataset. We used cross-validation with 5 folders on each training dataset to obtain the optimal value of the regularization constant. We used zero-one loss function in all cases.

First, we evaluated the descriptor on the segmented symbols of the FC database. There are 6 classes and there are 2 919 symbols in the train dataset and 1 861 in the test dataset. Since the symbols are already segmented, we do not have to define the class *no_match*. The accuracy was 98.9%.

The second database, on which we tested the descriptor, was NicIcon database of handwritten icons for crisis management by Niels et al. [10], which consists of 15 372 symbols (9 212 in the train dataset and 6 160 in the test dataset) of 14 classes (see Figure 7). We performed the writer independent experiment and obtained the accuracy 98.3%. It is way better than the result of Niels et al., which is 96.5%.

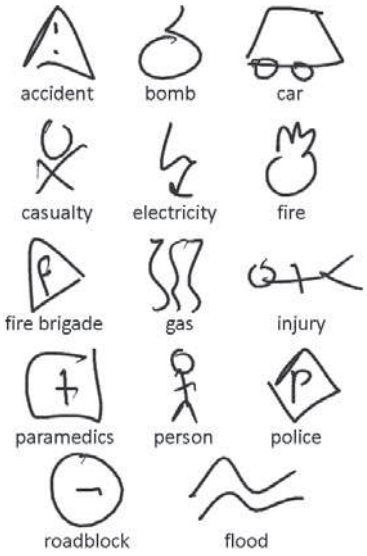


Figure 7. Symbol classes in the NicIcon database.

The last database was Unipen database of handwritten digits [6] which contains 4 990 digits (3 489 in the train dataset and 1 501 in the test dataset). In this experiment we obtained the accuracy 92.9%

which is significantly lower than in previous cases. The main reason probably is the fact that our descriptor is based on histograms and digits are very simple symbols consisting of only few points. Therefore, the histograms in our descriptor are rather dense.

5. Conclusion

We presented a new composition descriptor for describing hand-drawn graphical symbols. We trained the multiclass SVM classifier on various databases of graphical symbols, where the symbols were described with our descriptor. The result is outstanding for databases of more complex symbols. We achieved the accuracy 98.9% and 98.3% for FC database and NicIcon database, respectively. Experiments also showed that a poorer result is achieved on symbols consisting of just few points, which is the case of Unipen database of digits, where we achieved the accuracy 92.9%.

We used the descriptor to recognize symbols of flowcharts among symbol candidates, which are generated by strokes grouping. We generated the symbol candidates by grouping spatially close strokes together. This led to a segmentation, which contains a lot of candidates representing no symbol. Therefore, we learned the multiclass SVM classifier with negative examples. Moreover, we showed how the clustering of descriptors from the same class leads to better results. We were able to find 91.9% of the symbols in the FC database while we generate in average 8.8 times more symbol candidates than the number of symbols in a diagram.

In the future work, we plan to use the output of the segmentation as an input for the next stage of the pipeline, where relations between symbol candidates will be found. The goal will be to keep only the symbol candidates, which form a correct flowchart. Preliminary results based on modelling of relations and following optimization are promising and show that the number of candidates is not too large. We also plan to test the descriptor on more databases and possibly improve it to be more specialized on flowcharts, especially on filtering of the negative examples.

Acknowledgements

The authors were supported by the Grant Agency of the Czech Republic under the project P103/10/0783. The authors would like to thank V. Franc for his valuable advices in the field of machine learning.

References

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

[2] V. Babu, L. Prasanth, R. Sharma, and A. Bharath. Hmm-based online handwriting recognition system for telugu symbols. In *Ninth International Conference on Document Analysis and Recognition, 2007. ICDAR 2007.*, volume 1, pages 63 –67, sept. 2007. 2

[3] C. Bishop, M. Svensen, and G. Hinton. Distinguishing text from graphics in on-line handwritten ink. In *Ninth International Workshop on Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004.*, pages 142 – 147, oct. 2004. 2

[4] D. Blostein. General diagram-recognition methodologies. In *Selected Papers from the First International Workshop on Graphics Recognition, Methods and Applications*, pages 106–122, London, UK, UK, 1996. Springer-Verlag. 1

[5] V. Franc and V. Hlaváč. Statistical pattern recognition toolbox for Matlab. Research Report CTU–CMP–2004–08, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, June 2004. 3

[6] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proceedings of the 12th IAPR International Conference on Pattern Recognition, 1994. Vol. 2 - Conference B: Computer Vision amp; Image Processing.*, volume 2, pages 29 –33 vol.2, oct 1994. 6

[7] E. Indermühle, V. Frinken, and H. Bunke. Mode detection in online handwritten documents using BLSTM neural networks. In *ICFHR '12: Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition*, pages 302–307, 2012. 2

[8] L. B. Kara and T. F. Stahovich. Hierarchical parsing and recognition of hand-sketched diagrams. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM. 2

[9] M. Luqman, T. Brouard, and J.-Y. Ramel. Graphic symbol recognition using graph based signature and bayesian network classifier. In *10th International Conference on Document Analysis and Recognition, 2009. ICDAR '09.*, pages 1325 –1329, july 2009. 2

[10] R. Niels, D. Willems, and L. Vuurpijl. The NicIcon database of handwritten icons. In *11th International Conference on the Frontiers of Handwriting Recognition (ICFHR 2008)*, Montreal, Canada, August 2008. In press. 6

[11] S. Otte, D. Krechel, M. Liwicki, and A. Dengel. Local feature based online mode detection with recurrent neural networks. In *ICFHR '12: Proceedings*

of the 13th International Conference on Frontiers in Handwriting Recognition, pages 531–535, 2012. 2

[12] E. J. Peterson, T. F. Stahovich, E. Doi, and C. Alvarado. Grouping strokes into shapes in hand-drawn diagrams. In *AAAI'10*, 2010. 2

[13] M. I. Schlesinger and V. Hlaváč. *Ten Lectures on Statistical and Structural Pattern Recognition (Computational Imaging and Vision)*. Springer, 2002 edition, Mar. 2012. 1

[14] G. Shi and Y. Zhang. An improved svm-hmm based classifier for online recognition of handwritten chemical symbols. In *2010 Chinese Conference on Pattern Recognition (CCPR).*, pages 1 –5, oct. 2010. 2

[15] D. Tabernik, M. Kristan, M. Boben, and A. Leonardis. Learning statistically relevant edge structure improves low-level visual descriptors. In *International Conference on Pattern Recognition*, 2012. 5

[16] C. H. Teo, A. J. Smola, and Q. V. Le. Bundle Methods for Regularized Risk Minimization. *Journal of Machine Learning Research*, 11:311–365, 2010. 3

[17] X.-D. Zhou and C.-L. Liu. Text/non-text ink stroke classification in japanese handwriting based on markov random fields. In *Ninth International Conference on Document Analysis and Recognition, 2007. ICDAR 2007.*, volume 1, pages 377 –381, sept. 2007. 2

[18] B. Zhu and M. Nakagawa. Building a compact on-line mrf recognizer for large character set using structured dictionary representation and vector quantization technique. In J. E. Guerrero, editor, *ICFHR '12: Proceedings of the 13th International Conference on Frontiers in Handwriting Recognition*, pages 155–160, Los Alamitos, USA, 2012. IEEE Computer Society. 2