HAL
open science

Online flowchart understanding by combining max-margin Markov random field with grammatical analysis

Chengcheng Wang, Harold Mouchère, Aurélie Lemaitre, Christian Viard-Gaudin

To cite this version:

Chengcheng Wang, Harold Mouchère, Aurélie Lemaitre, Christian Viard-Gaudin. Online flowchart understanding by combining max-margin Markov random field with grammatical analysis. International Journal on Document Analysis and Recognition, 2017, 20 (2), pp.123-136. 10.1007/s10032-017-0284-8 . hal-01503366

HAL Id: hal-01503366

https://inria.hal.science/hal-01503366v1

Submitted on 7 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Noname manuscript No.
(will be inserted by the editor)

Online Flowchart Understanding by Combining Max-margin Markov Random Field with Grammatical Analysis[1]

Chengcheng Wang · Harold Mouchère · Aurélie Lemaitre · Christian Viard-Gaudin

Abstract Flowcharts are considered in this work as a specific 2D handwritten language where the basic strokes are the terminal symbols of a graphical language governed by a 2D grammar. In this way, they can be regarded as structured objects, and we propose to use a MRF to model them, and to allow assigning a label to each of the strokes. We use structured SVM as learning algorithm, maximizing the margin between true labels and incorrect labels. The model would automatically learn the implicit grammatical information encoded among strokes, which greatly improves the stroke labeling accuracy compared to previous researches that incorporated human prior knowledge of flowchart structure. We further complete the recognition by using grammatical analysis, which finally brings coherence to the whole flowchart recognition by labeling the relations between the detected objects.

1 Introduction

Sketched diagram is a powerful language that can help illustrating people's ideas. They contain self-explanatory symbols and can expand their styles freely. One typical example of such diagram is flowchart, which makes the illustrations of programs or structural objects very intuitive. With the emergence of electronic devices, the situations are becoming more common that people have to record their diagrams in their tablets and hopefully, the applications would facilitate the process of diagram parsing. Understanding and processing diagrams by machine learning techniques is also becoming a hot research topic, and fortunate enough, as a 2D handwritten language, the task of recognizing sketched diagram is similar to recognizing handwritten mathematical expressions and the latter can be referenced.

A traditional workflow of parsing 2D handwritten language is a trilogy: grouping strokes into symbols, recognizing symbols, and then analyzing structures [9]. We can follow this process in diagram recognition in a broad way, however, the properties differ from one specific 2D language to another. Cases are rare that one symbol is contained in another except square root operator in mathematical expressions, but it is common in flowchart that texts are contained in terminal nodes. Sketched diagrams also have a more flexible grammar in that they can almost expand their symbols in all directions, but in mathematical expressions, rules are settled so that there is less variability. From Figure 1 we can see that arrows in a flowchart can have various shapes, and can point to almost everywhere. Flowchart also cover a larger scope than mathematical expressions [9, 19]. All these add difficulties in sketched diagram recognition.

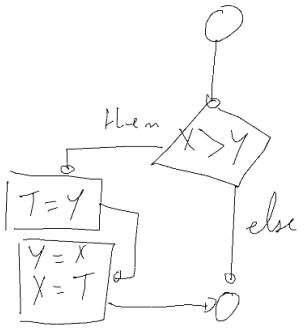
Although most of the 2D handwritten languages can be interpreted as the trilogy that has been mentioned above, recent researches tend to use a global method to avoid the propagation of errors, which decrease the recognition rate by feeding the misinterpreted symbols or structures into the following steps [9, 19].

Chengcheng Wang
Microsoft (China) Co. Ltd.
Suzhou, China
E-mail: ronwa@microsoft.com

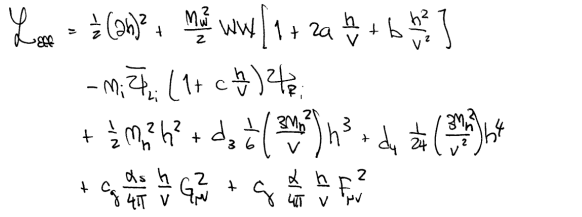
H. Mouchère and C. Viard-Gaudin
UBL/University of Nantes/LS2N, rue Christian Pauc, 44306 Nantes, FRANCE
Tel.: (+33) 2-40-68-30-82 E-mail: harold.mouchere@univ-nantes.fr / christian.viard-gaudin@univ-nantes.fr

Aurélie Lemaitre
IRISA - Université de Rennes 2, Campus de Beaulieu, 35042 Rennes Cedex, FRANCE
E-mail: aurelie.lemaitre@irisa.fr

[1] The final publication is available at Springer via http://dx.doi.org/10.1007/s10032-017-0284-8



(a) Example of flowchart



(b) Examples of mathematical expression

Fig. 1 Two 2D handwritten languages: sketched diagrams and mathematical expressions. The diagram has much flexibility while the mathematical expression follows a rather strict grammar.

Instead of enlarging the search space by generating several segmentations and several symbol hypotheses, our proposal is to guide the local recognition using neighborhood information. In this way, the problem is not solved by simply performing classification of individual component, that approach does not take into account the empirical fact that labels do not occur independently; instead, each label displays a strong conditional dependence on the labels of the surrounding components. To this end, document is arranged as a structured object, which can fit into an undirected graphical model, or more specifically, a Markov Random Field (MRF). A MRF that would incorporate local, neighborhood and global information would be a superior solution to solve symbol ambiguity problems. In addition, it is possible to train this model in a structured learning framework. Two alternatives exists for parameter learning: logistic loss based Conditional Random Field (CRF) approach and hinge-loss based structured SVM approach. CRF is almost intractable for general graphical models [17] [21], as shown in Figure 2. Meanwhile, several works [15] [30] showed that structured SVMs avoid the inference in the whole label space by a heuristic search in a primal form, making the learning tractable. This paves a new way of learning parameters of MRF, and due to its max-margin property of its loss function, the structural SVM can lead to a Max-Margin Markov Random Field (M3N).

In this work, we adopted structured SVM learning methods and MRF for the first time in the field of 2D handwriting recognition, leading to a novel statistical and structural information combined online flowchart recognition method, which is also applicable to other 2D handwriting problems. We build a M3N on the stroke level of the online flowcharts for stroke labeling. The parameters of M3N are learned using structured SVM. After stroke labeling, we use a grammatical description language extended from a previous work [18] to parse the structure of a given sketched diagram, thus obtaining the final recognized flowchart. In order to fully evaluate of the recognition process, the ground-truth of the flowchart dataset [2] have been completed by adding the object relations.

In the following sections, we first have a review of previous works on 2D handwritten language recognition and the application of structured learning, especially CRF and structured SVM, and how they inspired our handwriting recognition research. In the third section, we will describe the stroke labeling problem, in which a full description of our max-margin MRF building and training will be given. In the fourth section, we explain how the strokes labels are used in the syntactical analysis. The experiment will be given in the fifth section after a short description of the new version of the evaluation dataset. We will conclude the research in the sixth section.

2 Related researches

Flowchart can be viewed as one type of 2D handwritten languages. To enlarge the scope, we will review more generally 2D handwritten language recognition researches that complete symbol recognition and structure recognition simultaneously. Integrating neighborhood information in handwritten document understanding can be done globally by using strict language modeling like grammars or more locally through CRF or alternatively with structured SVM. Both kinds of approaches will be discussed in this section.

2.1 2D Handwritten language recognition

Significant research effort has been reported for recognition of math expressions in the recent years [19], in part due to online input devices becoming more popular. Zanibbi *et al.* [34] proposed *baseline structure tree* that parses the elements of a mathematical expression into a tree that fits natively its structure and then passed it to lexical parser and expression analyzer. The output of this three-pass system is an operator tree that can

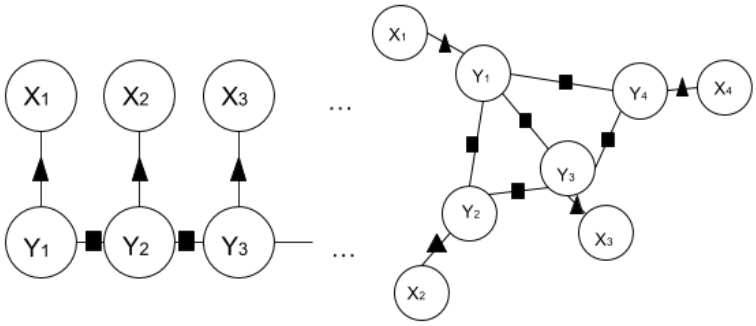


Fig. 2 Examples of graphical models. MRF and CRF share the same graphical models, but MRF are generative models which model the joint probability distribution, while CRF are discriminative models which model the conditional probability distribution. The black triangles on the edges are the unary factor nodes, while black squares on edges are the pairwise factor nodes.A linear chain model (left example) will be easier to train and will be used for inference with respect to a general model which contains cycles (right example).

be interpreted by computer. In [29], a minimal spanning tree (MST) is used to arrange all strokes into a graph and used symbol dominance information to set the weights of the tree so that they take into account the baseline of the mathematical expression. Symbol relation tree (SRT) is used in [26], it overcomes the baseline restrictions and build the tree using dominance symbol. They also used layered search to include symbol hypothesis and prune search space. These elegant solutions are well adapted for mathematical expressions, however, they might not be applicable to flowcharts or other sketched diagrams as the concepts *baseline* and *dominance symbol* are too specific. Another drawback of these approaches is their greedy decisions. In order to take the final decisions as later as possible, Awal *et. al.* [3] used a 2D dynamic programming algorithm symbol hypothesis generator, concatenating it with a Time Delayed Neural Network (TDNN) [31] symbol classifier. Besides, structure interpretation also suffers from ambiguity, which is caused by the ambiguous position of symbols and previous recognized symbols. Symbol relation tree is proposed in [26] to include all the symbol hypothesis and relation hypothesis. They used a heuristic measure to calculate the cost of a certain tree and they get the interpreted mathematical expression by best-first-searching for the interpretation with the lowest cost. Hence, this tree-style models may not be applicable to our diagram recognition needs.

There are no public flowchart or any other sketch diagram dataset until 2011 [2], and after that, many researches appeared to improve that first trial research. Lemaitre *et. al.* [18] analyzed the structure of the flowchart and described them in DMOS (introduced in section 4.1), a grammatical method for structured document analysis. This approach provides a global solution to the problem. The structure, symbol and stroke labels are recognized in one step. However, this unified recognition framework did not use any statistical learning methods. To improve this method, research combined statistical and structural information [8]. They used deformation scores to measure potential of candidate symbols so that they could dynamically switch to a preferable symbol in the structured search. However, the proposed approach did not cover simply all the possible variations to build a given symbol. For instance, the drawing of a rectangle can be done in a single stroke or can be done with 4 strokes, one side at a time. To deal with this variability, the structural approaches need to define one specific rule for specific case. Wu *et.al.* [33] proposed to recognize symbols in flowcharts by using shapeness, which is a property of whether a symbol has a *good* shape. They used a 256 dimensional feature and represented it compactly by using 16 INT64 numbers, calling it INT64 feature. The INT64 feature is used in a linear classifier to filter out symbols with *good* shape, then a neural network is used to achieve the final recognition. This research outperforms others due to its separation of processing of symbols according to whether they are having a regular appearance, this is too specific to the field of flowchart recognition: for mathematical expression recognition, there are less texts and symbols have regular shapes, and the position relationship between texts and symbols are more complicated.

Bresler *et.al.* proposed max-sum model in [6], which can be viewed as a structured model incorporating unary and pairwise relationships. Their max-sum model tried to maximize the global compatibility of symbols. They first extracted a bunch of symbol candidates (about 8.8 times the groundtruth symbol set size) and evaluated their combinations to find the most suitable symbols that fit in the flowcharts. They formulated the max-

sum model as a an Integer Linear Programming (ILP) [27] problem and solved it using IBM ILOG CPLEX[1]. In their following research [7], they refined their research by carefully dealing with text and arrow symbols: they first filtered out text by using Bi-directional Long-Short Term Memory neural network (BLSTM) based Bayesian classifiers then checked connection points of each symbol to decide the connecting arrows. They also published finite automata dataset to validate their research. One of the inspirations of their effort to our research is that their max-sum model is an attempt to include the global information in one model. Though achieving high recognition precision, filtering of text and their effort in deciding arrow type make their effort too specific to flowchart. Besides, the large amount of generated symbol hypothesis increased the inference effort in the max-sum model, and will always inevitably drop some true symbols.

2.2 Structured learning

Structured learning describes a wide range of problems that involve structured objects instead of scalar discrete or real values. By denoting $\mathcal{X}$ an input space, and $\mathcal{Y}$ an output space, the structured learning means fitting a function h to map from $\mathcal{X}$ to $\mathcal{Y}$ using a set of training samples in $(\mathcal{X} \times \mathcal{Y})$. By using the word *structured*, the output $\mathbf{Y}$ should be arranged in a structured manner, like trees or graphs. The task of stroke labeling in flowchart recognition corresponds to this description as the underlying strokes are arranged in a Markov random field. Here we have a review of previous researches that used CRF and M3N, two practical forms in structured learning [22].

As we have mentioned before, CRF is a powerful tool in describing regional context. Research utilized Bayesian CRF in their flowchart recognition [25]. They first modeled the distribution of parameters, pairwise probability and the inverse of partition function as Gaussian distributions, and then they repeatedly trained the model by minimizing Kullback–Leibler divergence between the new and the original models. This method is called expectation propagation. They did their research on text-free segment based flowchart recognition. However, Gaussian distribution is a strong assumption that may not applicable to pairwise potentials. Besides, their research did not continue with symbol recognition and their flowchart contains much fewer symbol types compared to the flowchart dataset mentioned in [2], only arrows and nodes are included.

[1] www.ibm.com/software/products/en/ibmilogcpleoptistud

A promising result using CRF in sketch diagram recognition is described in [11], though it is originally designed for text/nontext classification in handwritten documents. They build the CRF using a set of associative potential functions and interactive potential functions, the latter are built on a set of 5 relations, not limited to spatial, temporal and touching relations. In their later research [12], they further improved labeling accuracy by building a hierarchical tree-CRF, on the same handwritten document database, and they classify strokes to five classes, which is more specific than just text/nontext. They grouped and recognized symbols by building a MST on the recognized strokes of the same type. However, the hierarchical tree-CRF's performance is largely controlled by the clustering distance threshold, which is specific to the document and can largely affect CRF's performance according to their experiment.

On the other hand, M3Ns are showing promising performance in the field of image labeling. The property that CRF can be reduced to a structured SVM [15] [30] provides smooth transition from traditional research of CRF to max-margin MRF representation. One example is [24], which uses M3N to combine multiple local beliefs of superpixels and regional affinity. They also used global constraints to refine unary potentials. Structured SVM has also been applied in the field of object location [4], point matching [28] and pose esti-mation [13]. However, one common idea behind the construction of M3N is that unary potentials should describe the local belief based only on the local observation, while pairwise potentials, or even higher order potentials should describe regional interactions.

Regarding all the previous researches using MRF in handwriting recognition [35] [11] [12], MRF has shown to be a powerful structured model to describe neighborhood information. We discarded the idea of finding a tree-structure in the representation of sketched diagrams and resort to general graph MRF that may contain cycles to capture neighborhood relationship between strokes. Training by structured SVM learning algorithm is a natural solution to avoid the intractable learning problem in the partition function of CRF that may contains loops in underlying graph [17]. Training a M3N consists of learning the unary potentials and the pairwise potentials to incorporate local belief and neighborhood compatibility accordingly. With the labeled strokes, the grammar parsing will result in a legitimate, reasonable diagram. This is the first attempt to use structured SVM in 2D handwritten language recognition, and the combination of structural grammars and statistical recognition result gives a complete comparable referral on different recognition stages.

3 Stroke labeling using max-margin MRF

Suppose we have N diagrams, then we denote the stroke observation set as $\mathbf{X}^{(n)}$ for each diagram, which is indexed by n. We denote strokes $\left\{X_1^{(n)}, X_2^{(n)}, \ldots, X_{M_n}^{(n)}\right\} = \mathbf{X}^{(n)}$ that belong to the observation set. M_n is the number of strokes in an observation set. The topic of this section is stroke labeling problem, so each of the stroke observation has a corresponding label $Y_i^{(n)}$, which takes discrete labels such as *Arrow*, *Terminal* and so on, depending on the diagram. In the following, in case of no ambiguity, we simply define the observation set of a specific sample of diagram $\mathbf{X}$ and its corresponding label set as $\mathbf{Y}$, each strokes will be indexed by subscripts.

Classically, we use an energy function to describe the whole compatibility of the flowchart. It is composed of a set of unary energy functions and pairwise energy functions:

$$E(\mathbf{X}, \mathbf{Y}) = \left[\sum_i \epsilon(X_i, Y_i) + \sum_{<i,j>\in\mathcal{E}} \epsilon(X_i, X_j, Y_i, Y_j)\right] \tag{1}$$

In the energy function in equation (1), the unary energy function describes the local belief Y_i given only the isolated observation X_i. The pairwise energy function $\epsilon(X_i, X_j, Y_i, Y_j)$ provides regional compatibility. In many image segmentation researches, pairwise energy functions try to fix the local belief by smoothing regional labels given the regional observations. That energy function, under the assumption of Gibbs distribution [17], can be incorporated into a MRF:

$$\begin{aligned}\tilde{P}(\mathbf{X}, \mathbf{Y}) &= \exp\left[-E(\mathbf{X}, \mathbf{Y})\right] \\ &= \prod_i e^{-\epsilon(X_i, Y_i)} \prod_{<i,j>\in\mathcal{E}} e^{-\epsilon(X_i, X_j, Y_i, Y_j)} \\ &= \prod_i \Phi(X_i, Y_i) \prod_{<i,j>\in\mathcal{E}} \Phi(X_i, X_j, Y_i, Y_j)\end{aligned} \tag{2}$$

Here in formula (2), we use exponential form of potential function to ensure potential function are always positive, which is a requirement of MRF [17]. The inference problem is to solve $\mathbf{Y} = \underset{\mathbf{Y}}{\operatorname{argmax}}\, \tilde{P}(\mathbf{X}, \mathbf{Y})$, which can also be interpreted in the form of energy function as $\mathbf{Y} = \underset{\mathbf{Y}}{\operatorname{argmin}}\, E(\mathbf{X}, \mathbf{Y})$. In the following subsections, we will describe the feature selection, inference problem and learning problem separately.

3.1 Potential functions

The potential functions should be built to exhibit the relevant features from the raw data. Here we discuss the selected features for unary potential functions and pairwise potential functions, then describe the mathematical expressions.

3.1.1 Raw feature selection

Considering that diagrams drawn by different people have different scales, as a preprocessing step, we rescaled all the diagrams to a [-1, 1] bounding box and kept its height/width ratio. For the convenience of uniform feature extraction, we sampled each stroke to equal count of points with a constant spatial step. Our trial on the number of points per stroke showed that the denser the sampling the better the feature extraction ability, but performance is hard to improve after 30 points per stroke. For the balance between performance and processing time complexity, we set sampling point on each stroke to 30.

To extract raw features for unary potential functions $\Phi(X_i, Y_i)$, we calculate the Δx, Δy, $\cos\theta$, $\sin\theta$, $\cos\gamma$, $\sin\gamma$ on each sampling point as illustrated with Figure 3. The angle θ is the slope angle on the current point, and $\gamma_i = \theta_{i+1} - \theta_i = \Delta\theta$. This describes the curvature of that point. For the Δx, Δy, we normalize them by dividing the width and height of the bounding box of the flowchart separately. Altogether there are $30 \times 6 = 180$ features that describe a stroke, and we normalize them by dividing by the maximum absolute value of each component.

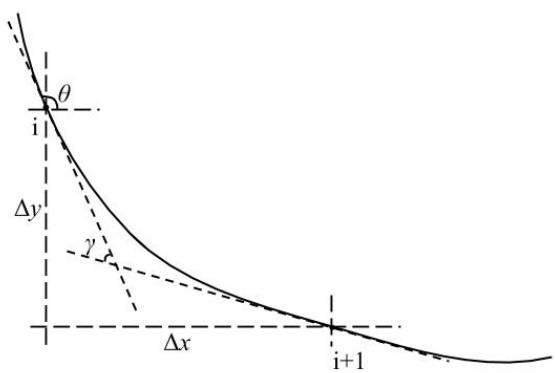
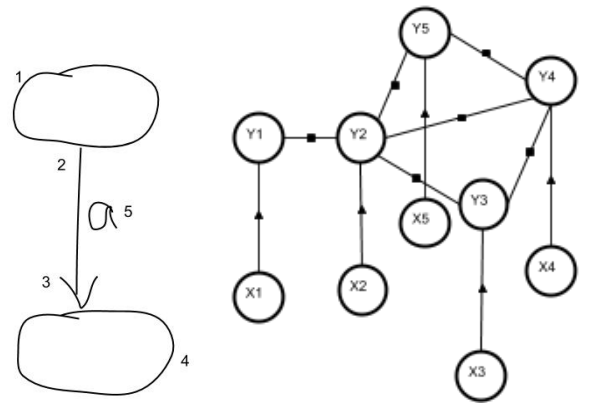


Fig. 3 This graph illustrates raw features for unary potentials. i and $i+1$ are two consecutive sampled points on a stroke, and we here showed features Δx, Δy, θ and γ for sampled point i.

The pairwise potentials $\Phi(X_i, X_j, Y_i, Y_j)$ should describe the regional compatibility of label Y_i, Y_j given

observation X_i, X_j. We focus on two types of neighborhood in pairwise function: temporal neighborhood and spatial neighborhood. Hence, from the time domain, X_i and X_{i+1} are considered as neighbors, so there is a potential function that describes their affinity. For the spatial neighborhood, we define a threshold distance. If the stroke pairs' minimal distance is below that threshold, they are considered as neighbors. To decide the threshold, we tested thresholds from 0.01 to 0.07 and evaluated stroke labeling accuracy and training time, and we took 0.03 as a preferable threshold in the following experiments. The detail of this spatial threshold decision can be referred in section 5. The pairwise potential function is undirected, which means $\Phi(X_i, X_j, Y_i, Y_j)$ is equivalent to $\Phi(X_j, X_i, Y_j, Y_i)$. The Figure 4 shows an example of sketch with its set of unary and pairwise potential functions.



(a) Example sketch diagram (b) Underlying Markov random field

Fig. 4 A sample sketch diagram and its corresponding MRF. Unary potential functions are represented by a solid triangle between X and Y, and pairwise potential functions are represented by a solid square between Ys. Those links indicate they are regarded as neighbors temporally or spatially.

We extract raw features for pairwise potentials by finding the displacement of each of the sampled points on one stroke to the point on another stroke that can offer the minimal distance. As shown in Figure 5, X_1 is the first sampling point on the left stroke, and from all the sampling points on the right stroke, Y_n has the minimal distance. The displacement from X_1 to Y_n is shown by a dotted arrow, and this displacement generates Δx and Δy as two pairwise raw features. For the sampling points Y_1 on the right stroke, we find the sampling point X_1 on the left stroke with minimal distance. We do this repeatedly on every sampling points of each pair of strokes. Altogether, there are $30 \times 2 \times 2 = 120$ pairwise raw features.

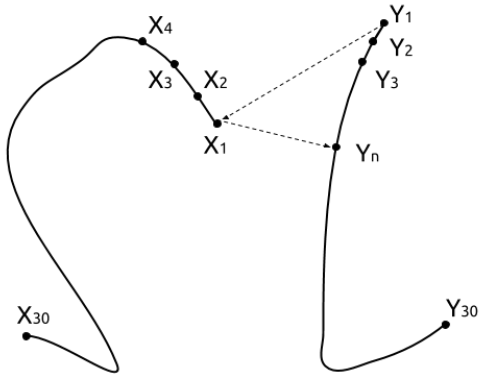


Fig. 5 This graph shows raw feature pairwise relationship. Every stroke is sampled to 30 points, $X_1, X_2, \ldots, X_{30}$ on the first stroke and $Y_1, Y_2, \ldots, Y_{30}$ on the second stroke. Only a selected number of points are presented. The features are the displacements vectors from every sampling point from one stroke to another, as shown by dotted arrow.

In many researches on analyzing online handwritten documents, researchers would chose manually selected and refined features to feed the classifiers [12]. We, however, simply selected very raw features as described above, and then use supervised classifiers to classify them to different types of stroke relationships, and the feature derived from the classification result is used as feature. In this way researchers avoided the effort to manually select different features according to different scenarios, making our method adaptable to fields beyond flowchart. However, domain specific raw features could have better results.

3.1.2 Mathematical formulation of potential functions

The unary potential function $\Phi(X_i, Y_i)$ can be formally defined according to equation (3).

$$\begin{aligned}\Phi(X_i, Y_i) &= \exp\left[-\sum_{l=1}^{L} \mathbb{1}_{Y_i=l} \sum_{k=1}^{K} w_{k,l} f_k(X_i)\right] \\ &= \exp\left[-\sum_{k=1}^{K}\sum_{l=1}^{L} w_{k,l}\left[\mathbb{1}_{Y_i=l} f_k(X_i)\right]\right]\end{aligned} \tag{3}$$

In the equation above, $\mathbb{1}_{Y_i=l}$ is the indicator function given the current label, and K is the number of features that are fed in this unary potential function. L is the number of possible labels. In this research, since unary potential functions are to describe the local confidence of a label, we should use a supervised classifier that gives confidence on final labels directly. Practically, all classifiers giving this kind of outputs could be used. Since raw features directly reflects the property of strokes which are of various shapes, and even same

type of stroke is having various shapes in flowchart, we choose ensemble methods to improve robustness of the feature extraction. We selected random forest [5], which uses averaging methods to combine different decision trees to prevent over-fitting, and GBDT [23], which is using boosting to combine several weak classifier to a strong one. The refined features are the probability of each label given the observation. Both classifiers give the probability of predicted class, and here we concatenated them to form refined feature. Since the $f_k(X_i)$ describe the probability of different labels based on two pre-trained classifiers, $K = 2 \times L$. The classifiers are trained on raw features of observations.

Following the idea of unary potential functions, the formulation of pairwise potential functions in equation 4 also used an indicator and a feature descriptor.

$$\begin{aligned}&\Phi(X_i, X_j, Y_i, Y_j)\\&= \exp\left[-\sum_{l_1=1,l_2=1}^{l_1,l_2\in L} \mathbb{1}_{Y_i=l_1,Y_i=l_2} \sum_{k}^{K} w_{l_1,l_2,k} f_k(X_i, X_j)\right]\\&= \exp\left[-\sum_{k}^{K} \sum_{l_1=1,l_2=1}^{l_1,l_2\in L} w_{l_1,l_2,k} \left[\mathbb{1}_{Y_i=l_1,Y_i=l_2} f_k(X_i, X_j)\right]\right]\end{aligned} \tag{4}$$

That feature descriptor should describe local properties that reflect local relationship of two neighboring strokes, thus completing local compatibility description. We continue using random forest and GBDT, taking pairwise raw features and classify them into given stroke relationships. For within-symbol strokes, the symbol label is used as stroke relationship label, and for inter-symbol strokes, stroke relationship inherits from symbol relationship (for a concrete example, section 5.1 describes possible symbol relationships for flowchart database). By adding "no relationship" for other situations, all possible stroke relationships are included.

Though there are choices for a more sophisticated and advanced classifier, we simply disregarded the abundant choice for other researchers to push our boundary further. Whatever classifier one would use, the automatic feature extraction process from raw features to the classified class enables one to use this identical process in their own handwriting recognition task is one contribution in our work, which greatly facilitated the transplanting of our system to other problems.

3.2 Model inference

Inference problem has been a main focus of probabilistic graphical model research. Here in this research, we adopt Alpha-Expansion-Fusion algorithm [14]. This is a graph-cut based algorithm so it is faster than belief propagation. Besides, this algorithm adopts Quadratic Pseudo-Boolean Optimization (QPBO) so that its energy does not have to be submodular.

3.3 Parameter learning

In the structured SVM framework, a discriminant function $F(\mathbf{X}, \mathbf{Y}; \mathbf{w})$ exists describing the compatibility of label $\mathbf{Y}$ and observation $\mathbf{X}$. The dot product of the feature vector $\mathbf{f}(\mathbf{X}, \mathbf{Y})$ and parameter $\mathbf{w}$ can give a good description of $F(\mathbf{X}, \mathbf{Y}; \mathbf{w})$, in which the parse *feature function* reflects the property of the structured objects itself. As formulas (3) and (4) indicate, potential function can be reorganized in the format as $\exp[-\mathbf{w} \cdot \mathbf{f}(X, Y)]$, so by aggregating (2), (3) and (4) we can get the feature vector representation and parameter vector $\mathbf{w}$, expressed in formula (5). Inference problem in the form of discriminant function can be expressed as $\mathbf{Y} = \underset{\mathbf{Y}}{\operatorname{argmax}}\, F(\mathbf{X}, \mathbf{Y})$. The MRF expression formula (2) is not a linear representation of features, however, minimizing the energy function is equivalent to maximizing the structured learning discriminant function, as shown in formula (5):

$$F(\mathbf{X}, \mathbf{Y}) = -E(\mathbf{X}, \mathbf{Y}) = \mathbf{w} \cdot \mathbf{f}(\mathbf{X}, \mathbf{Y}) \tag{5}$$

A solution to generalize normal SVM training to structured outputs is presented in [15]. All the things to do in training is to find a cutting-plane, described by $\mathbf{w}$, that separates right labels from those inferred by the models but with wrong labels. In this research, we use margin-scale problem formulation and solve the problem by using one-slack structural SVM learning algorithm [16]. The problem can be formulated as follows:

$$\min \frac{1}{2}||\mathbf{w}||^2 + C\xi, \tag{6}$$

with the constraints

$$\begin{aligned}&\frac{1}{N}\sum_{i=1}^{N}\Big\{\Delta(\mathbf{Y}^{(i)}, \bar{\mathbf{Y}}^{(i)})\\&-\mathbf{w}\cdot\Big[\mathbf{f}(\mathbf{X}^{(i)}, \mathbf{Y}^{(i)}) - \mathbf{f}(\mathbf{X}^{(i)}, \bar{\mathbf{Y}}^{(i)})\Big]\Big\} \le \xi\\&\qquad\qquad \forall(\bar{\mathbf{Y}}^{(1)}, \bar{\mathbf{Y}}^{(2)}, \ldots, \bar{\mathbf{Y}}^{(N)}) \in \mathcal{Y}^N.\end{aligned} \tag{7}$$

In formula (7) $\bar{\mathbf{Y}}^{(i)}$ is the predicted labels for sample i, while the true labels is denoted as $\mathbf{Y}^{(i)}$. The structural loss $\Delta(\mathbf{Y}^{(i)}, \bar{\mathbf{Y}}^{(i)})$ defined in this loss function specifies the margin between true label and the

incorrect label in our max-margin MRF training. The one common form of structural loss is Hamming distance between true labels and predicted labels, and we continue use that in this paper. The difficulty of structural SVM is that the label space $\mathcal{Y}^{(n)}$ may be tremendously large: it takes L^{M_n} possible label combinations, so the constraints in the QP may be very large, which could not be handled easily. For this problem, [16] proposes N-slack and one-slack structural SVM training algorithm that uses a heuristic search strategy to add constraints dynamically. The main idea is to get the most violated instance in each iteration and to add that into constraints set $\mathcal{W}$, replacing the constraints in (7), then solve the QP in primal space. The iteration will stop when the constraint set remains unchanged after a certain iteration. The majority of constraints are behind the most violated one, so it is unnecessary to add all the constraints. For the completeness of this paper, we repeat the algorithm in **Algorithm 1**.

Algorithm 1 Training Structured SVM by using margin-rescaling One-Slack formulation

Require: $S = \left[(\mathbf{X}^{(1)}, \mathbf{Y}^{(1)}), \ldots, (\mathbf{X}^{(N)}, \mathbf{Y}^{(N)})\right], C, \epsilon$

$\mathcal{W} \leftarrow \emptyset$
repeat
  $(\mathbf{w}, \xi) \leftarrow$ Solving QP, replacing constraints (7) with $\mathcal{W}$
  for all $i \in [1, \ldots, N]$ **do**
    $\bar{\mathbf{Y}}^{(i)} \leftarrow \underset{\bar{\mathbf{Y}}^{(i)} \in \mathcal{Y}}{\operatorname{argmax}} \left[\Delta(\mathbf{Y}^{(i)}, \bar{\mathbf{Y}}^{(i)}) + \mathbf{w} \cdot \mathbf{f}(\mathbf{X}^{(i)}, \bar{\mathbf{Y}}^{(i)})\right]$
  end for
  if $\frac{1}{N}\sum_{i=1}^{N} \Big\{\Delta(\mathbf{Y}^{(i)}, \bar{\mathbf{Y}}^{(i)}) - \mathbf{w}\left[\mathbf{f}(\mathbf{X}^{(i)}, \mathbf{Y}^{(i)}) - \mathbf{f}(\mathbf{X}^{(i)}, \bar{\mathbf{Y}}^{(i)})\right]\Big\} > \xi + \epsilon$ **then**
    $\mathcal{W} \leftarrow \mathcal{W} \cup \left(\bar{\mathbf{Y}}^{(1)}, \bar{\mathbf{Y}}^{(2)}, \ldots, \bar{\mathbf{Y}}^{(N)}\right)$
  end if
until $\mathcal{W}$ unchanged during iteration

4 Grammatical analysis of handwritten flowcharts

Once the isolated strokes have been labeled, we propose to combine the results using a grammatical description. This second step does ensure the global consistence of the recognition. The presented system is an extension of the grammatical analysis from the previous works [18] and [8]. In this former solutions, the analysis was based only the set of atomic elements in data: the full strokes and the straight-lines (segments) within the strokes. Using these terminals, their relative layout, and the *a priori* lexical knowledge about flowcharts the syntactical analysis recognizes the symbols. Compared to [18] this new work adds two main propositions: using pre-recognized strokes from a statistical approach (the MRF in our case) and producing the relationship labels between the recognized symbols.

4.1 The DMOS approach

The DMOS (Description and MOdification of the Segmentation) method [10] is a grammatical method for structured documents analysis. The using of the grammatical language EPF (Enhanced Position Formalism) enables a syntactic and structural description of the content of a document. From the description in EPF of a model of documents, it is possible to compile automatically an analyzer which will be used to process the documents.

The DMOS engine is based on the Prolog language. It means that a document is described with a set of rules. For example the following rule allows to build a quadrilateral object using four segments (`TERM_SEG`) locally named `S1`, ..., `S4`.

```
quadrilateral Q::=
   TERM_SEG S1 &&
   AT(end S1) && TERM_SEG S2 &&
   AT(end S2) && TERM_SEG S3 &&
   AT(end S3) && TERM_SEG S4
```

The `AT(.)` operator allows to add constraints on the next used terminals. For example `AT(end S1) && TERM_SEG S2` means that `S2` should be at the end of the the segment `S1`. Without these constraints, a quadrilateral would be recognized for each set of four segments whatever their layout. Using this rule, it is possible to build a top rule to recognize a complex document based on several quadrilaterals. As DMOS is based on a Prolog solving engine, the analyzer can try and test, using backtracking, all combinations of segments to satisfy the main rule. The order of the rules in the grammar can have a big impact on how efficient is this search. Indeed some predicates can have to enumerate a lot of candidates. To optimize this search, the current version of DMOS is able to build and save intermediate objects (like quadrilaterals in the previous example) and then to use them as new terminals in the explanation of a full document.

4.2 The Flowchart analysis

The grammar processes the strokes in three main steps:

1. build all boxes found in the documents using [8],
2. label the boxes and connect them with arrows,
3. use the remaining strokes to build text areas.

The second step follows a simple flowchart grammar: start with a *terminator* or a *connection* symbol, then continue with an *arrow* to the next box; each box (*terminator*, *connection*, *process* or *data*) is followed by an *arrow* which ends on a box; *decision* boxes are followed by two arrows; ...

The next rule illustrates in a simplified EPF language how an *arrow* follows a *process* box. This recursive predicate builds a list of symbols starting by a *process* named `Pr` (of course the real rules deal also with the other types of boxes) and an *arrow* named `Arr`. The first step is to check if the box `Pr` satisfies the graphical conditions to be a *Process*. The predicate `ProcessCond` can be seen as a binary classifier of *Process* boxes. Then the *arrow* candidates are selected in an area closed to the box `Pr`. At the end of the arrow, another box among the previously found boxes should exist. If these conditions are satisfied the link between the arrow `Arr` and the next box `B` is saved and the recursion is called to build the remaining parts of the flowchart. If the new box `B` is also a *Process*, the same version of `recursiveDiag` will be used to look for the new arrow. The second part of the predicate is an example of ending predicate: the last box should be a *connection* box.

```
recursiveDiag [Pr, Arr|Remaining]::=
   ProcessCond Pr && Pr == Process &&
   AT(closeTo Pr) && anArrow Arr &&
   AT(end Arr) && Box B in savedBoxes &&
   save Arr B &&
   recursiveDiag [B|Remaining]

recursiveDiag [LastBox]::=
   LastBox == Connection
```

It should be noticed that the real grammar should accept flowcharts which are not perfectly drawn. For example in Figure 10 an arrow is missing to connect the "Input n,m" box and the "r= n%m" box.

The grammar is designed to force the explanation of all available strokes. Thus if some strokes are remained at the end of the analysis and if they can not be included in a *text* area in the last step, then the predicate fails and thanks to the backtracking, another solution is explored.

In the previous work [18] the system was based only on the syntactic description of the document based on the relative positions of the strokes. In [8] some statistical information based on the boxes shapes are used to first detect the boxes and then, using the same structural grammar as in [18], build the flowchart.

In this work the labels obtained from the MRF are used to improve the syntactical analysis at three points:

- box labeling: in the first step of the analysis, if all strokes of a box have the same label, then the box will have this label, otherwise criteria from [8] are used;
- arrow ending: during the third step, if a remaining stroke labeled as an *arrow* is close to a detected arrow, then they are merged;
- text detection: during the third step, the text areas start with a stroke labeled as *text* and then they are expanded horizontally with the unused strokes.

5 Experiments

We performed our sketch diagram parsing experiments on the updated publicly available dataset Flowchart Dataset [2].

5.1 Data description

The flowchart dataset consists of handwritten flowcharts of various complexity. Some are complex algorithms with descriptive texts while some are just simple and straightforward sequential operations. The used samples are the same as in the already published Flowchart dataset [2] but the ground-truth has been updated. The symbols in the flowchart dataset are divided into 7 classes: Data, Terminator, Process, Decision, Connection, Arrows and Text. Strokes inherit their belonging symbol label. Compared to the initial version, the ground-truth contains the symbol segmentation and labels but also their relationships in the flowchart. Thus, the latest version includes three directional relationships between symbols: i) Associated Text between node or arrow and their associated texts; ii) Source and iii) Target, which clarify the relationships between arrow and linked nodes. This opens the possibility for researches to utilize and recognize these relationships. The flowcharts have been collected using digital pens from university students of the research laboratory. This dataset is split into train set and test set. However, we further manually divided the training set into three folds to perform 3 fold cross-validation for parameter tunning. We divide the training set in such a way that each fold has no flowchart writers overlapped with the writers in other folders. We sacrificed the equality of each divided size for no overlapped writer, to avoid the situation where the writers' writing habits contributes to the recognition result on validation set. For the detailed statistics of flowchart dataset, please refer to Table 1.

Table 1 Flowchart database statistical overview. 10 writers of the Test part appear also in the Train part.

	Writers	Patterns	Flowcharts	Strokes	Symbols
Training	30	14	248	23355	5540
Training fold 1	11	14	76	6896	1688
Training fold 2	11	14	83	7898	1861
Training fold 3	8	14	89	8561	1991
Test	15	14 other	171	15696	3791

5.2 Parameter tuning by cross-validation

Our model has several hyper-parameters to be decided, and that can be done by cross-validation using our divided 3-fold training set. Here in this subsection, we briefly explain how to decide the spatial distance threshold mentioned in 3.1.1. Spatial distance threshold would affect the graphical model's structure, and the smaller the spatial distance is, the sparser the graphical model would be, which may effect the training and inference time, and stroke labeling performance. In each round of training, we select one fold as validation set, and train on the combination of the other 2 folds. Then we take the average on each fold as current parameter's performance measure. In this research, we use `pystruct` [20] back-ended by `OpenGM` [1] to perform structured SVM parameter learning and M3N inference.

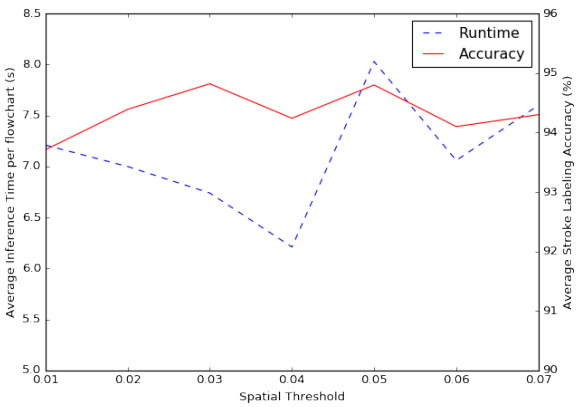


Fig. 6 This experiment shows the result of tuning of spatial distance threshold by 3-fold cross-validation. Two metrics are to decide the final choice of spatial distance threshold: inference time on per flowchart, and stroke labeling accuracy.

Fig 6 shows the spatial distance threshold's effect on model inference time and stroke labeling accuracy. For stroke labeling accuracy before 0.03, it is increasing steady. This is due to the fact that with larger spatial distance threshold, the more connections exist for vertices on graphical models, the richness of connection provides more information of the flowchart. However, the accuracy appears to be unstable after that because expanded edges are providing a redundancy, which is having a side effect on graphical model learning and inference. By selecting two peak values of stroke labeling accuracy and drop 0.05 which is having a longer inference time, we finally selected 0.03 as threshold value.

We use similar steps for determination of other parameters, such as the minimal number per leaf and the number of decision trees in random forest classifier, and maximal number of leaves of GBDT. Since there are too many parameters to do grid-search, we selected one type of parameters, then settle down others to perform parameter grid search by cross-validation. Then we switch to another type of parameter, until we have tuned all the parameters. Finally, the random forest has 80 trees and 14 as minimal samples per leaves and GBDT has 25 trees with a maximal 31 leaves per tree to form the unary potential function feature refiner; for the pairwise potential function feature refiner, random forest has 80 trees and 3 minimal samples per leaves and GBDT has 30 trees with a maximal 127 leaves per tree to form the refined feature generator.

5.3 Stroke recognition

After the determination of parameters, we train our model on the whole training set and perform test on test set. The first stage of our flowchart recognition system is to use M3N to determine stroke labels. Due to the randomness of random forest and GBDT, we launched 5 experiments to recognize flowchart. The average recognition result can be referred to Table 3, and we selected the experiment that has median recognition rate to illustrate the detail below. There are 7 classes of symbols in flowchart, so strokes are classified into 7 classes. Here we show the confusion matrix in Figure 7 and precision, recall and F1 score of our model in Table 2:

Table 2 Stroke labeling result for each class and all together before and after grammatical analysis (GA). P for precision, R for recall. (%)

	Before GA			After GA		
	P	R	F1	P	R	F1
Text	97.88	98.42	98.15	96.81	99.27	98.03
Arrow	93.93	92.29	93.10	98.18	90.49	94.18
Process	90.17	94.08	92.08	93.13	94.69	93.91
Terminator	95.03	77.13	85.15	89.29	78.48	83.53
Connection	91.87	83.70	87.60	83.22	91.85	87.32
Data	90.00	91.56	90.77	93.28	94.59	93.93
Decision	90.44	93.82	92.10	95.69	92.53	94.08
All	95.71	95.70	95.67	96.19	96.18	96.14
Accuracy	95.70			96.18		

Our M3N model is using GBDT and Random Forest as raw feature extractor, and we also tested their stroke

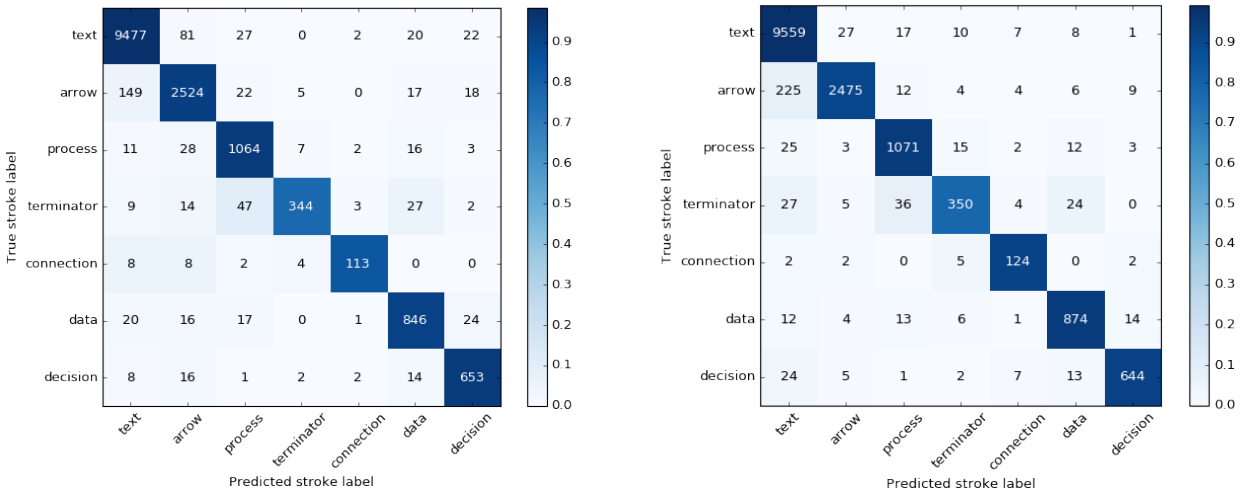


Fig. 7 Confusion matrix for strokes labeling result on test set. The left one is the result after M3N inference but before grammatical analysis. The right one is after grammatical analysis. Each row of the matrix contains all the strokes of the groundtruth class, and each column contains all the strokes of the predicted class. The denser each block is on the diagonal, the higher the recall rate.

recognition rate along with M3N combined stroke recognition rate. From Table 3 we can see that when using a standalone classifier to classify strokes, the classifier does not use neighborhood information and it is keeping at a recognition rate a little above 80%. By combining it with M3N, we observed a significant improvement of stroke recognition rate. This validated our belief that MRF is suitable for modeling the 2D handwritten language.

From the confusion matrix before grammatical analysis, *terminator* class is having the lowest recall rate. That is because terminator is having a similar shape with *process*. *Process* is actually a rectangle while *terminator* is a rounded rectangle, there's no doubt why many strokes belonging to *terminator* are classified to *process*. Besides, they can also be misclassified to *arrow* because *arrow* is of various shape itself. Another interesting point is that *text* have the highest recall rate, that's mainly because *text* has its own style, they would cluster together and in a much smaller scale. Also, the biased number of training samples of different strokes are having a side-effect on recognition result: stroke classes with few samples are more likely to be misclassified to other classes.

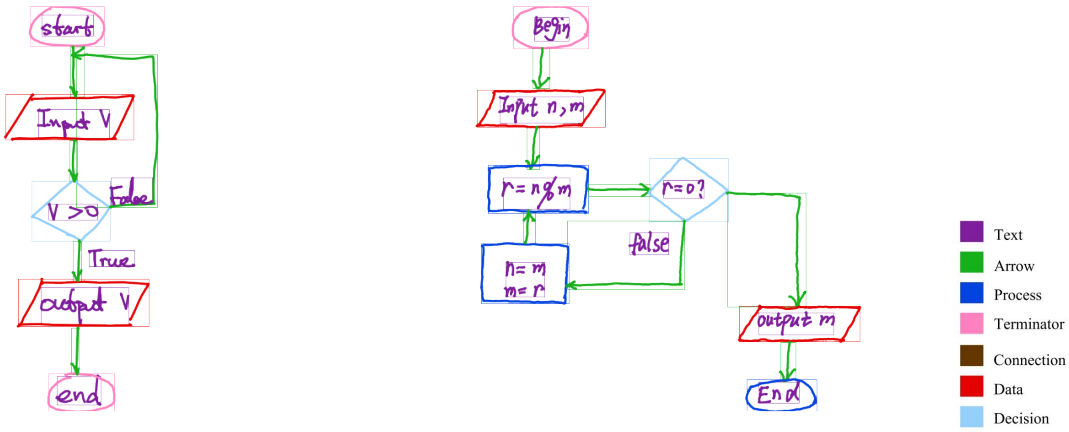
Fortunately, with grammatical analysis, this biased sample problem is eased. From the right confusion matrix, which is the result after grammatical analysis, we observe an obvious decrease of error recognition case on *arrow*, *process* and *data*. However, there's an increase in the error where strokes are labeled as *text*. The class *text* is hard to encoded in grammatical rules so our grammatical analysis process fails to process it along with other shapes, and has to be left till last. Overall, grammatical analysis improved the labeling result from our statistical model from 95.70% to 96.18%, proving it to be helpful in statistical model improvement. We also performed a dual-side *t-test* on stroke recognition rate compared to the state-of-the-art stroke recognition rate [32], and we are having a confidence above 90% that current model is superior than [32]. Considering that research [32] is also conducted by us, by using MRF learned with structured SVM, they are of same origination and they are all significantly superior to flowchart recognition methods using other models.

Table 3 Stroke labeling result and final symbol recognition recall on the test set of flowchart dataset (%). Considering the randomness from Random Forests and GBDT, our experiment launched for 5 times, with mean and unbiased standard error recorded. This table also listed some other researches on the same dataset.

	Stroke Reco. Rate	Symbol Reco. Rate
Lemaitre *et. al.* [18]	91.1	72.4
Carton *et. al.* [8]	92.4	75.0
Bresler *et. al.* [7]	95.2	82.8
Wu *et. al.* [33]	94.9	83.2
Wang *et. al.* [32]	95.8	**84.3**
GBDT only	80.87 ± 0.07	-
Random Forest only	81.45 ± 0.10	-
GBDT+RF+M3N inference	95.70 ± 0.65	-
GBDT+RF+M3N inference+GA	$\mathbf{96.16 \pm 0.34}$	80.69 ± 0.85

5.4 Symbol and graph level recognition

When the stroke recognition results are available, we can process the grammatical structural analysis. This analysis enables to build both objects based on strokes, and logical relationships between recognized symbols.

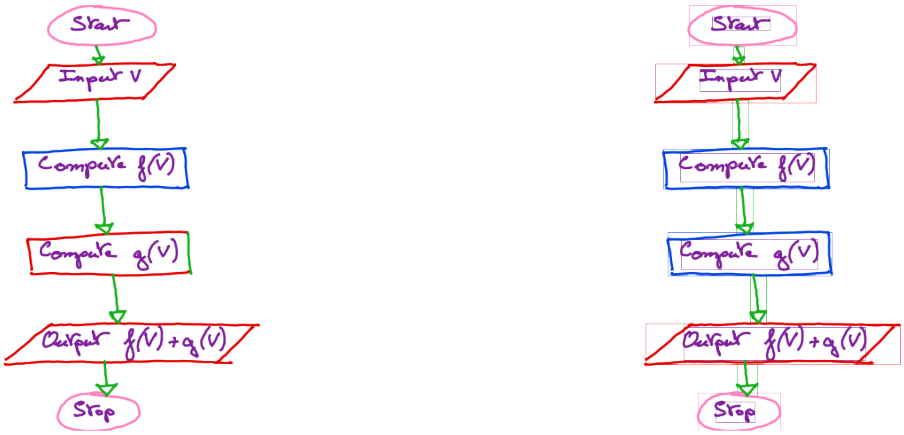


(a) Flowchart recognized without error.

(b) Correctly segmented flowchart but with one recognition error: one *terminator* is recognized as *process*.

(c) Legend

Fig. 8 Typical examples of flowchart recognition, with max-margin MRF stroke labeling and grammatical analysis. The strokes are colored to indicate their recognition result. The legend of Figures is shown in 8(c).



(a) An incorrectly recognized flowchart, recognized with max-margin MRF only without grammatical analysis.

(b) Same flowchart recognized after grammatical analysis.

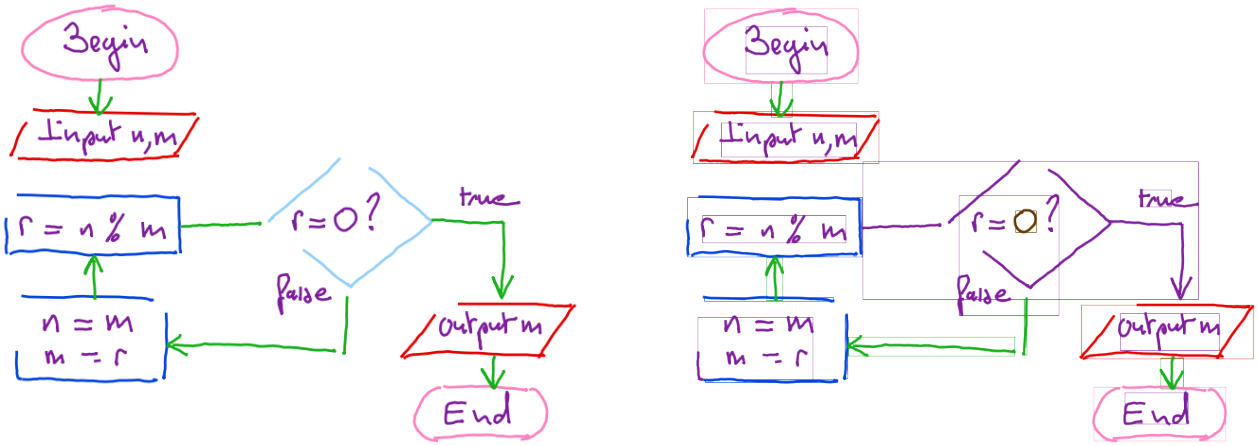
Fig. 9 A flowchart's recognition result in two stages: 9(a)for after max-margin MRF stroke labeling and 9(b)after stroke labeling and grammatical analysis. Grammatical description would eliminate some stroke labeling errors, for example the *process* is confused with *data* symbol before grammatical analysis. Most of this type of errors are corrected after grammatical analysis step.

The novelty of our experiments is that the database now includes relationships in the ground truth. Consequently, we can evaluate structural correctness in our graph level recognition analysis. Our graphical level evaluation is done using the tools mentioned in [19].

Table 4 gives object level recognition result, specifically symbol segmentation results and symbol recognition results. Traditional recall and precision metrics compute the ratio of correctly recognized symbols to groundtruth symbols and correctly recognized symbols to all recognized symbols respectively. We can notice a higher rate for symbol segmentation than for sym-

Table 4 Symbol segmentation and recognition result after grammatical analysis (%)

	Precision	Recall	F1
Text	71.06	78.93	74.79
Arrow	72.89	76.89	74.84
Process	84.95	89.95	87.38
Terminator	78.21	77.83	78.02
Connection	85.18	92.74	88.80
Data	92.63	89.80	91.19
Decision	82.86	82.07	82.46
Object overall	76.13	80.85	78.42
Segmentation	78.21	83.06	80.57



(a) A correctly labeled flowchart using max-margin MRF.

(b) The same flowchart recognized after grammatical analysis, which leads to error.

Fig. 10 An incorrectly recognized flowchart compared to the state before grammatical analysis. 10(a) is correctly labeled. The *arrow* and the *decision* symbol are all recognized as *text*, while letter *o* is recognized as *connector*. This is due to the undetected *decision* box (as in [8], Fig. 6). Missed boxes can not be recognized.

bol recognition because a correctly recognized symbol requires both a valid segmentation and a valid symbol label.

Table 5 Graph level recognition on test set, considering all 171 flowchart instances

	Rate(%)	Correct count
Correct symbol segmentation	19.88	34
Correct symbol recognition	12.87	22
Correct symbol + structure	5.85	10

Table 5 shows the recognition results at the global flowchart level. A diagram is considered correctly recognized only when all of its symbols are correctly segmented, correctly labeled and the structure is correctly parsed . These three requirements come in an order that a later requirement must meets a prior requirement, so in table 5 we give recognition rate under these three more and more strict requirements. It can be seen that the correctly recognized diagrams are relatively few considering all the requirements. However, we need a more detailed analysis of the causes of errors.

In Figures 8, 9 and 10 we have shown four examples of labeled flowcharts: 8(a) is a correctly recognized flowchart, 8(b) is a correctly segmented but incorrectly labeled sample, and Figures 9 and 10 show incorrectly recognized flowcharts. From the examples we can have some insight of the source of the errors. For 8(b), the only error occurs with a *process* box recognized as *data*. This implies that though grammatical analysis ensure a valid interpretation, a better flowchart recognition result can be achieved by using stronger local raw feature extractor. Compared to some researches that filtered text first as [7], our performance is satisfying. The goal of the additional grammatical structural analysis is to recover as much as possible these errors. However, the quality of the results will depend on the strength of the rules that are hard-coded and relies heavily on the designer experience. This is why, in some cases, the initial probabilistic model would give better results than the post-processed one. Such a situation is illustrated in Figure 10 where from the correctly labeled flowchart obtained with the max-margin MRF, cf. Figure 10(a), an incorrect interpretation is then produced applying the grammar rules, cf. Figure 10(b). To improve global recognition rate, higher order MRF should be considered and shape information incorporated grammar parser, like [8], can be used to avoid labeling part of the node symbol into strokes.

6 Conclusion

In this work a mixed strategy is proposed for recognizing online handwritten flowcharts. From one hand, we develop MRF models trained using structured SVM to label the flowcharts at the stroke level. On the other hand, a syntactic approach is used to parse the results of the stroke labeling and to produce symbols with their relationships in consistence with grammar rules. In a sense, an implicit and an explicit grammar modeling are achieved with this solution. We validated our approach on a flowchart dataset that now contains a complete

ground-truth, including logical order, and that is available for experiments.

Furthermore, this symbol recognition method can support additional improvements. First of all, we can balance differently the local and the global constancies. In this direction, MRF can support higher order potential functions. This could be helpful for producing better segmentation results. Secondly, a deeper integration of our MRF model and grammatical model could be studied to use a probabilistic parsing method to alleviate the dominant effect of the stroke labeling. In further research we will investigate the potential of MRF more deeply to make it more compatible to global consistency.

References

1. Bjoern Andres, Thorsten Beier, and Jörg H. Kappes. OpenGM: A C++ library for discrete graphical models. *ArXiv e-prints*, 2012.
2. Ahmad-Montaser Awal, Guihuan Feng, Harold Mouchere, and Christian Viard-Gaudin. First experiments on a new online handwritten flowchart database. In *IS&T/SPIE Electronic Imaging*, pages 78740A–78740A. International Society for Optics and Photonics, 2011.
3. Ahmad-Montaser Awal, Harold Mouchère, and Christian Viard-Gaudin. A global learning approach for an online handwritten mathematical expression recognition system. *Pattern Recognition Letters*, 35:68–77, 2014.
4. Matthew B Blaschko and Christoph H Lampert. Learning to localize objects with structured output regression. In *Computer Vision–ECCV 2008*, pages 2–15. Springer, 2008.
5. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
6. Martin Bresler, Daniel Pruša, and Václav Hlavác. Modeling flowchart structure recognition as a max-sum problem. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1215–1219. IEEE, 2013.
7. Martin Bresler, Truyen Van Phan, Daniel Pruša, Masaki Nakagawa, and Václav Hlavác. Recognition system for on-line sketched diagrams. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 563–568. IEEE, 2014.
8. Céres Carton, Aurélie Lemaitre, and Bertrand Couasnon. Fusion of statistical and structural information for flowchart recognition. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1210–1214. IEEE, 2013.
9. Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition*, 3(1):3–15, 2000.
10. Bertrand Coüasnon. Dmos, a generic document recognition method: Application to table structure analysis in a general and in a specific way. *International Journal on Document Analysis and Recognition, IJDAR*, 8(2):111–122, 2006.
11. Adrien Delaye and Cheng-Lin Liu. Contextual text/non-text stroke classification in online handwritten notes with conditional random fields. *Pattern Recognition*, 47(3):959–968, 2014.
12. Adrien Delaye and Cheng-Lin Liu. Multi-class segmentation of free-form online documents with tree conditional random fields. *International Journal on Document Analysis and Recognition (IJDAR)*, 17(4):313–329, 2014.
13. Vittorio Ferrari, Manuel Marin-Jimenez, and Andrew Zisserman. Progressive search space reduction for human pose estimation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.
14. Alexander Fix, Aritanan Gruber, Endre Boros, and Ramin Zabih. A graph cut algorithm for higher-order markov random fields. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1020–1027. IEEE, 2011.
15. Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.
16. Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.
17. Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
18. Aurélie Lemaitre, Harold Mouchère, Jean Camillerapp, and Bertrand Coüasnon. Interest of syntactic knowledge for on-line flowchart recognition. In *Graphics Recognition. New Trends and Challenges*, pages 89–98. Springer, 2013.
19. Harold Mouchère, Richard Zanibbi, Utpal Garain, and Christian Viard-Gaudin. Advancing the state of the art for handwritten math recognition: the crohme competitions, 2011–2014. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 173–189, 2016.
20. Andreas C. Müller and Sven Behnke. pystruct - learning structured prediction in python. *Journal*

of Machine Learning Research, 15:2055–2060, 2014.

21. Sebastian Nowozin, Peter V Gehler, and Christoph H Lampert. On parameter learning in crf-based approaches to object class image segmentation. In *Computer Vision–ECCV 2010*, pages 98–111. Springer, 2010.
22. Sebastian Nowozin and Christoph H Lampert. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.
23. Artificial Intelligence Group of Microsoft Research Asia. Github - microsoft/lightgbm, 2016. https://github.com/Microsoft/LightGBM.
24. Deli Pei, Zhenguo Li, Rongrong Ji, and Fuchun Sun. Efficient semantic image segmentation with multi-class ranking prior. *Computer Vision and Image Understanding*, 120:81–90, 2014.
25. Yuan Qi, Martin Szummer, and Thomas P Minka. Diagram structure recognition by bayesian conditional random fields. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 191–196. IEEE, 2005.
26. Taik Heon Rhee and Jin Hyung Kim. Efficient search strategy in structural analysis for handwritten mathematical expression recognition. *Pattern Recognition*, 42(12):3192–3201, 2009.
27. Dan Roth and Wen-tau Yih. Integer linear programming inference for conditional random fields. In *Proceedings of the 22nd international conference on Machine learning*, pages 736–743. ACM, 2005.
28. Alex J Smola, Julian J Mcauley, and Tibério S Caetano. Robust near-isometric matching via structured learning of graphical models. In *Advances in Neural Information Processing Systems*, pages 1057–1064, 2009.
29. Ernesto Tapia and Raúl Rojas. Recognition of online handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In *Graphics recognition. Recent advances and perspectives*, pages 329–340. Springer, 2003.
30. Choon Hui Teo, Alex Smola, SVN Vishwanathan, and Quoc Viet Le. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2007.
31. Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.
32. Chengcheng Wang, Harold Mouchère, Christian Viard-Gaudin, and Lianwen Jin. Combined segmentation and recognition of online handwritten diagrams with high order markov random field. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016.
33. Jie Wu, Changhu Wang, Liqing Zhang, and Yong Rui. Offline sketch parsing via shapeness estimation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1200–1206. AAAI Press, 2015.
34. Richard Zanibbi, Dorothea Blostein, and James R Cordy. Recognizing mathematical expressions using tree transformation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(11):1455–1467, 2002.
35. Bilan Zhu and Masaki Nakagawa. On-line handwritten japanese characters recognition using a mrf model with parameter optimization by crf. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 603–607. IEEE, 2011.