Offline Sketch Parsing via Shapeness Estimation

Jie Wu[1,*] **, Changhu Wang**[2,†]**, Liqing Zhang**[1]**, Yong Rui**[2]
[1]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Microsoft Research, Beijing, P. R. China

Abstract

In this work, we target at the problem of offline sketch parsing, in which the temporal orders of strokes are unavailable. It is more challenging than most of existing work, which usually leverages the temporal information to reduce the search space. Different from traditional approaches in which thousands of candidate groups are selected for recognition, we propose the idea of shapeness estimation to greatly reduce this number in a very fast way. Based on the observation that most of hand-drawn shapes with well-defined closed boundaries can be clearly differentiated from non-shapes if normalized into a very small size, we propose an efficient shapeness estimation method. A compact feature representation as well as its efficient extraction method is also proposed to speed up this process. Based on the proposed shapeness estimation, we present a three-stage cascade framework for offline sketch parsing. The shapeness estimation technique in this framework greatly reduces the number of false positives, resulting in a 96.2% detection rate with only 32 candidate group proposals, which is two orders of magnitude less than existing methods. Extensive experiments show the superiority of the proposed framework over state-of-the-art works on sketch parsing in both effectiveness and efficiency, even though they leveraged the temporal information of strokes.

1 Introduction

Drawing diagrams and flowcharts is an important means of conveying information and structure in various domains. However, it is still a challenging problem for a computer to understand the structure of diagrams and flowcharts. Current flowchart softwares like Visio and SmartDraw still rely on the traditional point-click-drag style of interaction. It is highly desired if we have a practical recognition system that can parse and recognize the structure of diagrams/flowcharts in various domains. Although there is an increasing interest in building systems that can automatically interpret hand-drawn diagrams/flowchart, many challenges still remain in terms of recognition accuracy, extensibility to different input data types, and real-time performance, as shown in the example sketches in Fig. 1.

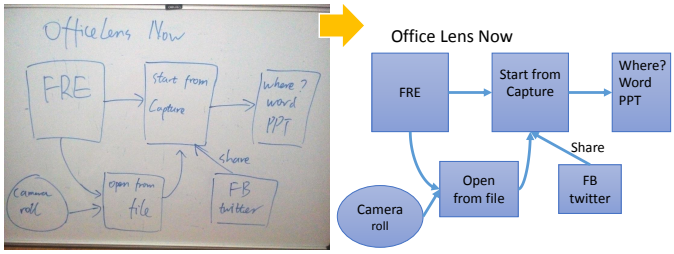

Figure 1: Illustration of offline sketch parsing. The goal of offline sketch parsing is to parse and recognize the shapes and text in a real-world image of hand-drawn diagrams taken from a mobile phone (left), which could be further converted to a powerpoint file with structured shapes (right).

To recognize the symbol in a sketch, a main-stream strategy is to first generate candidate stroke groups, followed by the recognition of each group, named as *selection-recognition* here. Theoretically, if we search for all possible candidate groups, the candidate group number for recognition is 2^N, which grows exponentially with the stroke number N. Therefore, to improve the efficiency and avoid the exponentially growing recognition cost, different kinds of constraints are leveraged to simplify the problem of full sketch recognition. Typical constraints include explicit cue constraint, temporal constraint, and spatial constraint. For explicit cue constraint, some works require explicit cues of users, such as button clicks or pauses, to separate each object [Hse and Richard Newton, 2005]. In this way, the full sketch recognition problem is reduced to isolated sketch recognition [Kara and Stahovich, 2004; Ouyang and Davis, 2009; Paulson and Hammond, 2008; Sun *et al.*, 2012b], which has the assumption that there is only one shape in the sketch. Temporal (Spatial) constraint means to only search among the groups of temporally (spatially) neighboring strokes [Bresler *et al.*, 2013; Lemaitre *et al.*, 2013; Carton *et al.*, 2013; Sun *et al.*, 2012a], to reduce the number of candidate groups.

[*]Jie Wu performed this work while being an intern at Microsoft Research Asia.

[†]Corresponding author.

Most existing work on full sketch recognition leveraged the temporal order information of strokes to reduce the search space, and thus are actually *online sketch parsing*. In this work, we focus on the problem of *offline sketch parsing*, which refers to methods without temporal constraints. In the real-world scenario, the input may be an image of sketch taken by mobile devices such as cell phone or Google Glass, as shown in Fig. 1, in which the temporal orders of strokes are unavailable. On the other hand, the assumption that a shape is drawn using temporally-close strokes may not be always right. E.g., [Sezgin and Davis, 2008] proposed a time-based model to tackle the problem that a sketch might contain interspersed drawing; [Wu *et al.*, 2014] also showed that in some shapes with correction/editing, the strokes may not be temporally close.

Relatively few work has been done in offline sketch parsing. An early work [Shilman and Viola, 2004] presented a framework to spatially group and recognize shapes and texts, which was however applied on simple cases, and might suffer from efficiency problem when applied for complex flowcharts. Some work focused on the recognition of printed flowcharts in patent images [Yu *et al.*, 1997; Futrelle and Nikolakis, 1995], where some prior knowledge of printed shapes such as existence of sharp points of some shapes (e.g., rectangle) was leveraged, and thus cannot be directly applied to free hand-drawn strokes.

Existing work on online sketch parsing might suffer from some problems if directly adapted for offline sketch parsing by removing temporal constraint. On the one hand, the strokes extracted from an image will be much more noisy than online sketched data and thus bring more strokes. On the other hand, without temporal order of strokes, more candidate groups will be generated, which will not only damage the recognition results, but also slow the recognition process. For example, using the *selection-recognition* strategy, for a simple flowchart drawing with 460 strokes extracted, 22597 candidate groups were generated only using spatial constraint, and the parsing process cost more than 100 seconds, which is far from satisfactory.

In this paper, we propose a practical framework for offline sketch parsing using the *selection-recognition* strategy. The basic idea is that, instead of directly recognize candidate stroke groups like in other methods, we first detect a small number of stroke groups that represent good shapes in a fast way, and then recognize these groups. We observed that most of hand-drawn shapes with well-defined closed boundaries could be distinguished from non-shapes if normalized into a very small size, as shown in Fig. 2. Based on this observation, the concept of *shapeness* as well as an efficient shapeness estimation algorithm is proposed to estimate whether a group of strokes represent a good shape. To speed up the shapeness estimation process, we introduce a very compact but effective feature representation to represent each stroke group. This feature is better able to handle the range of visual and stroke-level variations of sketched shapes in freehand drawings. To avoid frequently extracting features for the same stroke which might belong to multiple groups, we further propose an efficient feature extraction method by pre-calculating the major information of each stroke.

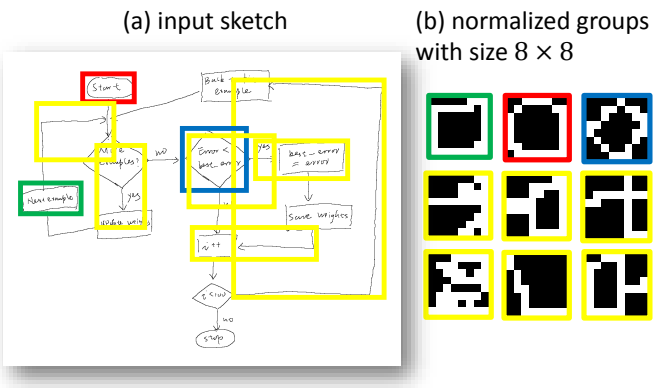

Figure 2: Illustration of the motivation of shapeness estimation. Although the candidate stroke groups of shapes (red, green, blue boxes) and non-shapes (yellow boxes) present huge variation in the sketch space (a), when resized to a very small scale (b), they share strong correlations respectively.

Based on the proposed shapeness estimation, we present a three-stage cascade framework for offline sketch parsing, including 1) shapeness estimation, 2) shape recognition, and 3) sketch parsing using domain knowledge. Extensive experiments are conducted on both shapeness estimation and sketch parsing. The shapeness estimation technique in this framework greatly reduces the number of false positives, resulting in a 96.2% detection rate with only 32 candidate group proposals, which is two orders of magnitude less than existing methods. The proposed framework also outperforms state-of-the-art works on sketch parsing in both effectiveness and efficiency, although counterpart algorithms leveraged the drawing orders of strokes to make the problem much easier.

2 Shapeness Estimation

As aforementioned, the traditional *selection-recognition* approach in state-of-the-art methods [Bresler *et al.*, 2013; Lemaitre *et al.*, 2013; Carton *et al.*, 2013] might suffer from both effectiveness and efficiency problem in the face of offline sketch parsing if without the temporal information. Thus, we propose the concept of shapeness as well as an efficient shapeness estimation algorithm to reduce the number of candidate stroke groups in a fast way.

2.1 Motivation and Problem Statement

Shapeness

We observe that the strokes in a flowchart image can be divided into two categories: 1) appearance-variant shapes which are similar in global, but might differ in local appearance such as rectangles, circles and diamonds, etc. 2) geometric-variant strokes including connectors and texts, which have large geometric deformation such as arrows, curves, lines, texts, etc. We find that the former *good* shapes can be clearly differentiated from the latter ones or non-shape stroke groups, if resized to a small scale, as shown in Fig. 2.

Based on this observation, we propose the concept of *shapeness*, to indicate how much a group of strokes likes a

good shape. Our strategy is to develop a fast estimation approach to calculate the shapeness of every candidate stroke group, and the ones with top scores will be reserved for further shape recognition.

Challenges

To guarantee the accuracy of sketch parsing, the shapeness estimation algorithm should achieve a high recall with only a few output groups. On the other hand, the algorithm should be very efficient, since it is designed to speed up the whole process. Denoting N as the number of strokes extracted from an image, theoretically there will be 2^N candidate groups to process. Although we can add the spatial constraint to reduce this number, the number of groups is still very large, making the efficiency problem more challenging.

Shapeness vs. Objectness

One idea is to leverage the objectness estimation technique in computer vision to solve the shapeness estimation problem, if we consider the *good* shapes as a type of *objects*.

Objectness is usually represented as a value to reflect how likely an image region covers an object of any category. Objectness estimation algorithms [Alexe *et al.*, 2012; Uijlings *et al.*, 2013; Cheng *et al.*, 2014] tried to detect (thousands of) bounding boxes to cover as many objects as possible.

However, as shown in experiments, to achieve a high recall (95%+), a state-of-the-art objectness estimation method [Cheng *et al.*, 2014] needs to output two orders of magnitude more groups than the proposed method, even with a loose criteria widely used by objectness estimation works. The major reason is that, candidate regions are usually represented by bounding box in objectness estimation. It can be considered as a very strong *spatial constraint* to exclude all non-rectangle regions. It can significantly speed up the process, and reasonable for natural images, but will not work for hand-drawn sketches where shapes and texts might interact in many ways. For example, it cannot separate a shape with the text inside it. On the other hand, it is not easy to directly generalize the algorithms to non-rectangle regions.

In this work, instead of using bounding-box to reduce the search space, we use the stroke groups as the basic elements to estimate the shapeness. Although it can solve the problems that objectness estimation algorithms meet, this will generate more candidate groups to calculate shapeness. Thus, an efficient algorithm is highly desired.

2.2 Shapeness Estimation Algorithm

Given a collection of strokes extracted from an image, we first generate a large number of candidate stroke groups to guarantee a high recall. Then, the compact INT64 features of these groups are extracted in a fast way, followed by the shapeness estimation of each group. Finally, only a small number of stroke groups with top scores are reserved.

Candidate Group Generation

As aforementioned, supposing the image has N strokes, there will be 2^N possible groups. We leverage the spatial constraint to reduce the search space. We first calculate the distance between every stroke pair s_i and s_j, denoted as $Dist(i, j)$, which is defined as the minimum distance of points between s_i and s_j. The Breadth First Search (BFS) is leveraged to generate candidate groups that satisfy the following conditions: (1) the stroke number of a group is less than a threshold TH_{depth}; (2) for any two strokes s_i, s_j in the group, $Dist(i, j) < TH_{dist} \times L_{diag}$, where L_{diag} is the median diagonal length of all strokes and TH_{dist} is a parameter; (3) the candidate groups with aspect ratio larger than TH_{aspect} are removed.

In this work, TH_{depth}, TH_{dist}, and TH_{aspect} were set to 6, 0.04, and 7, which were learnt from the training set.

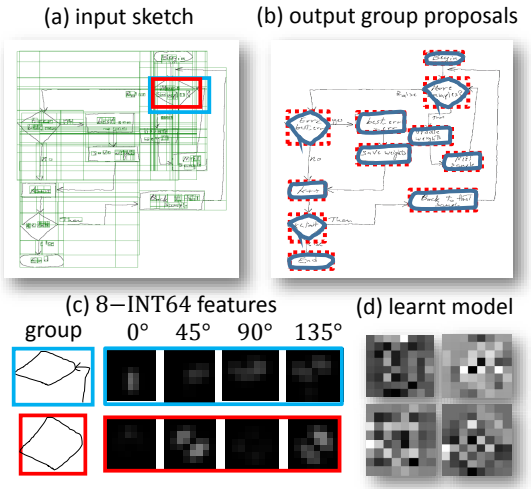

Figure 3: Overview of the shapeness estimation algorithm. Given a collection of strokes (a), the algorithm generates a small number of *group proposals* (b). First, we generate a lot of candidate stroke groups, as shown in green boxes in (a). Then, for each group, the INT64 feature is extracted for each group (c). Finally, a linear model (d) was learnt to estimate the shapeness of a group based on the INT64 feature. Only the groups with top values are kept as output proposals for further analysis (b).

Compact Feature Representation

We need an effective but compact feature representation to 1) handle stroke-level variations of hand-drawn shapes, and 2) facilitate the fast feature extraction. We first present a 256D feature to represent each stroke group, and then introduce how to convert it to the compact INT64 representation.

We follow the idea of the 720-dimension visual feature [Ouyang and Davis, 2009] to design a 256D feature. First, we normalize the candidate stroke group by translating its center of mass to the origin, and scaling it horizontally and vertically to 8×8 pixels so it has unit standard deviation in both axes. This is necessary to handle scale and translation variants of hand-drawn shapes. Then, four 8×8 orientation feature images are extracted corresponding to four orientations. Furthermore, we apply Gaussian smoothing to reduce sensitivity to local distortions, resulting in four 8×8 feature images ($4 \times 8 \times 8 = 256D$), as shown in Fig. 3(c). Each pixel in the images is an unsigned int (uint) in [0, 255].

To speed up the feature extraction and testing process, we

further convert the 256D feature to the INT64 feature. For the four feature images in a 256D feature, we quantify each pixel value to 16 values, using four bits: $abcd$, $(a, b, c, d \in \{0, 1\})$. Therefore, we can use 16 INT64 to represent the feature of a candidate group, which was named as *INT64* feature.

Fast Feature Extraction

As aforementioned, the number of candidate stroke groups might be very large, so the key challenge is how to efficiently extract the INT64 features for all the groups. Although each stroke belongs to multiple candidate groups, because the normalization step mentioned in last subsection makes the aspect ratio and position of a stroke change a lot in related groups, we cannot directly reuse the features of a single stroke. Therefore, instead, our solution is to avoid duplicate operations on the same stroke as much as possible, when extracting features for related groups.

Let's first check the operations to extract the feature from a candidate group with k strokes. The length of the ith stroke is n_i. Naive method to extract features is to 1) scan each point in the group to calculate the center of gravity (c_x, c_y), 2) scan each point to get the standard deviation (σ_x, σ_y) using the (c_x, c_y), and 3) scan again to calculate the feature. The estimated cost is $3 \times k \times n_{avg}$ operations, where n_{avg} is the average length of one stroke. Suppose there are totally M stroke groups resulted from N strokes in the image, and we have $M >> N$. So the cost for all groups is $3kMn_{avg}$.

In order to speedup the process, we first scan all N strokes for only one time to obtain all necessary statistics of each stroke, and then calculate c_x, c_y, and $\sigma^2_{\{x,y\}}$ in each group.

- **Pre-calculate statistics for each stroke.** Let $p_x(j)$ and $p_y(j)$ denote the coordinates of the jth point in a stroke. For the ith stroke, we calculate the point number $n(i)$, center of gravity point $c_{\{x,y\}}(i)$, sum of point values along both axes $sum_x(i) = \sum_j p_x(j), sum_y(i) = \sum_j p_y(j)$, sum of squared values in both axes $sumsqr_x(i) = \sum_j p_x(j)^2, sumsqr_y(i) = \sum_j p_y(j)^2$. Pre-calculating these values in advance enables us to efficiently calculate the center of gravity, and the deviation of each group.
- **Calculate statistics for each group.** For each group, the center of gravity can be calculate by

$$c_{\{x,y\}} = \frac{\sum_{i=1}^{k} c_{\{x,y\}}(i) \times n(i)}{\sum_{i=1}^{k} n(i)}. \quad (1)$$

The standard deviation can be calculated by

$$\sigma^2_{\{x,y\}} = \frac{1}{n}\sum_{t=1}^{n}(p_{\{x,y\}}(t) - c_{\{x,y\}})^2 \quad (2)$$

$$= \text{avg}(p_{\{x,y\}}(t)^2) - 2c_{\{x,y\}}\ \text{avg}(p_{\{x,y\}}(t)) + c^2_{\{x,y\}}, \quad (3)$$

where $\text{avg}(p_{\{x,y\}}(t)^2) = \frac{\sum_{i=1}^{k} sumsqr_{\{x,y\}}(i)}{\sum_{i=1}^{k} n(i)}$, and $\text{avg}(p_{\{x,y\}}(t)) = \frac{\sum_{i=1}^{k} sum_{\{x,y\}}(i)}{\sum_{i=1}^{k} n(i)}$. Therefore, (c_x, c_y) and σ_x, σ_y can be calculated by scanning over k strokes, instead of scanning over each point.

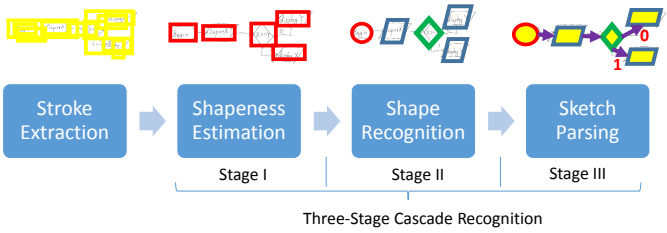

Figure 4: The pipeline of the proposed offline sketch parsing framework.

Thus, in the feature image generation step, we reduce the operation number from $3kMn_{avg}$ to kMn_{avg}, which is three times faster.

Finally, we use standard methods to conduct Gaussian smoothing and quantization to get the final INT64 feature for each group, which will be not detailed here due to space limitation.

Learning Shapeness Measurement

For efficiency, we learn a linear model w using linear SVM:

$$s =< w, g > +b, \quad (4)$$

where s, g are shapeness score, and the INT64 features of the candidate group. The features of stroke groups of groundtruth *good* shapes, and random sampled candidate groups are used as positive and negative training samples respectively. The learnt model, i.e., four 8×8 feature images, are shown in Fig. 3(d). We can see that, the outer parts of the model images are lighter (i.e., larger value) then the inner parts, showing that the boundaries of shapes are emphasized in deciding whether the group likes a generic shape.

Testing an INT64 Feature

For a new INT64 feature g, we can calculate its shapeness score $s =< w, g > +b$. Finally, we only keep the stroke groups with scores higher than a predefined threshold, e.g., 0 in this work, for further shape recognition.

3 Offline Sketch Parsing Framework

In this section, we introduce a three-stage cascade framework for offline sketch parsing. For an image with flowcharts, we could leverage standard stroke extraction techniques [Kovari, 2007; LHomer, 2000] to extract strokes. Given the strokes, our goal is to 1) group the strokes into distinct objects/shapes; 2) assign a label to strokes in each group from a label set, e.g., $L = \{$*text, rectangle, parallelogram, diamond, eclipse, circle, arrow*$\}$ for flowcharts. The grouping of text, shapes, connectors, and noises has potential to be used in further processing and analysis.

The overview of our framework is shown in Fig 4. First, we adopt shapeness estimation to detect possible *good* shapes using only a small number of proposals. Second, an isolated sketch recognizer is applied to recognize the detected shapes. Finally, domain knowledge is leveraged to parse and group each stroke into shapes, text, and connectors.

3.1 Stage I: Shapeness Estimation

Given a collection of strokes, we adopt the proposed shapeness estimation method to get the candidate groups. Please see the previous section for technical details of this part.

3.2 Stage II: Shape Recognition

For each candidate stroke group detected as a shape in Stage I, we leverage a more accurate isolated sketch recognition algorithm to predict the shape type. Any well-performed classifier can be used here to recognize hand-drawn shapes. In this work, we adopt 1NN classifier, using the original 720 visual features in [Ouyang and Davis, 2009]. Each group is scored as the distance of feature vectors between the input shape and the shape in training set. After scoring each group, we sort these groups by their scores, followed by Non-Maximal Suppression (NMS). A group will be removed if 1) it shares a same stroke with a group with a better (smaller) score; or 2) its bounding box totally contains (or is totally contained by) a group with a better (smaller) score.

3.3 Stage III: Sketch Parsing using Domain Knowledge

After Stage I and II, the appearance-variant shapes can be recognized. Next, we try to group and parse other strokes, including the text inside shapes, text outside shapes, arrows, and so on. First, we group the text inside the recognized shapes. Second, the connectors between the recognized shapes are recognized. Finally, we leverage the spatial relationship to group the rest of strokes that are spatially close to each other.

Group Text inside Recognized Shapes

We first use the Graham Scan algorithm [Anderson, 1978] to compute the convex hull for every recognized shape. Then, the strokes inside the convex hull will be grouped together. A handwritten text recognition algorithm could be used to further analyze the grouped strokes, but here we do not leverage it. Our method can be served as a pre-processing module for text recognition.

Recognize Connectors

We use the Depth First Search (DFS) to search among groups that are spatially connected to two different recognized shapes. The DFS starts from groups with one stroke, one endpoint of which is (spatially) close to a recognized shape. Then our algorithm gradually adds a stroke that is close to the current group. When any two endpoints of the current group are found to be close to two different recognized shapes, we use a rule-based connector recognizer to recognize whether the group is an arrow or curve. This implementation is in similar spirit with [Hammond and Paulson, 2011]. A benefit of the rule-based recognizer is that the original connector can be automatically replaced by the formal connector, because the parameters are needed to perform beautification such as the endpoints of the connector are also calculated by the rule-based recognizer, as shown in the example in Fig. 1. Finally, we use the endpoint positions of connectors to determine whether it connects to any shape.

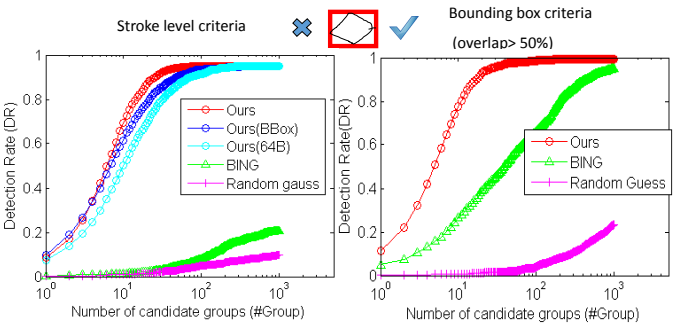

Figure 5: The shape detection recall curves of different methods using two criteria: stroke level criteria, and bounding box criteria.

Group the Rest of Strokes

After recognizing shapes, connectors, and text inside shapes, we continue to group the rest of strokes as noisy strokes or other kinds of patterns. We simply group any two strokes s_i, s_j together if their distance $Dist(i, j)$ is less than a predefined threshold.

4 Experiments

In this section, we evaluate the proposed shapeness estimation algorithm and the offline sketch parsing framework. The experiments were conducted on the FC Dataset [Lemaitre *et al.*, 2013], a widely used benchmark dataset on sketch parsing. It contains 419 flowcharts of various complexity (different patterns) written by 31 writers, in which 171 were used as testing data. These flowcharts contain 5,000+ symbols, including connection, terminator, data, decision, arrow, process, and text. The drawing orders of the strokes in each sketch are also provided. It should be noted that, to simulate the problem of offline sketch parsing, the proposed algorithm did not leverage this information, although all compared sketch parsing algorithms leveraged it to make the problem much easier.

4.1 Shapeness Estimation

We first evaluate the proposed shapeness estimation algorithm. Two evaluation criteria were adopted:

- **Stroke level criteria:** The predicted group is considered as correct if the strokes in this group are exactly the same as the ground truth group. This criteria is adopted by most state-of-the-art sketch parsing work, and thus we considered it as the default criteria in the rest of experiments.
- **Bounding box criteria:** If the overlap between the bounding box of a groundtruth group and that of the predicted group is larger than 50%, it is considered as a correct match. This criteria is widely used in object detection literature in computer vision. We also adopted this criteria because it is the criteria of the counterpart algorithm BING [Cheng *et al.*, 2014].

It is obvious that the stroke level criteria is much stricter than the bounding box criteria. We compare the proposed algorithm (Ours) with four algorithms and variants: 1) BING:

Method	Accuracy (%)	
	Stroke	Symbol
[Bresler *et al.*, 2013]	83.8	74.3
[Lemaitre *et al.*, 2013]	91.1	72.4
[Carton *et al.*, 2013]	92.4	75.0
Ours(without shapeness)	86.2	70.3
Ours	**94.9**	**83.2**

Table 1: Comparison on sketch parsing. Note that all compared methods leveraged temporal information to make the recognition easier. Moreover, [Bresler *et al.*, 2013] removed all the text from the flowchart dataset, making the problem simpler.

Class	Accuracy (%)			
	Stroke		Symbol	
	Stat + struct	Ours	Stat + struct	Ours
Connection	80.3	73.3	82.4	73.4
Terminator	69.8	**91.6**	72.4	**90.6**
Data	84.3	**87.6**	80.5	78.5
Decision	90.9	89.7	80.6	78.9
Process	90.4	**91.8**	85.2	**88.3**
Arrow	83.8	**87.4**	70.2	**80.3**
Text	97.2	**98.8**	74.1	**86.0**
Total	92.4	**94.9**	75.0	**83.2**

Table 2: Comparison of accuracy of all classes between Stat + struct [Carton *et al.*, 2013] and our method.

a state-of-the-art objectness detection method [Cheng *et al.*, 2014], 2) Random Gauss: randomly generate windows and consider strokes within the windows as candidate groups, 3) Ours(BBox): replace our normalization method with a simpler one, i.e., directly resizing the strokes into a fixed size bounding box, 4) Ours(64D): replace our INT64 feature with another feature, i.e., resizing the group of strokes into an 8×8 binary image, and using the binary image as the feature. Ours(64D) can be considered as an implementation of BING's feature for binary input.

Fig. 5 shows the detection recall curves of shapes, with the number of output groups changing.

Comparison of features. The superiority of Ours over Ours(BBox) in Fig. 5 shows the necessity of the normalization part in the feature extraction. The reason is that, the bounding box resizing technique is overly sensitive to artifacts like long tails at the ends of strokes or stray ink [Ouyang and Davis, 2009]. On the other hand, the Ours(64D) in Fig. 5 is very similar to the feature in BING, showing that the INT64 feature is better able to handle the stroke-level variations of hand-drawn shapes. The time cost of shapeness estimation using the INT64 feature is 0.36s.

Comparison with BING. Our method greatly outperformed BING, a state-of-the-art objectness estimation method, for both criteria. For stroke level criteria, the results of BING were quite bad, because it is bounding-box based method, and thus cannot separate a shape with the text inside. This shows that the window proposals in objectness estimation cannot well handle the stroke level labeling.

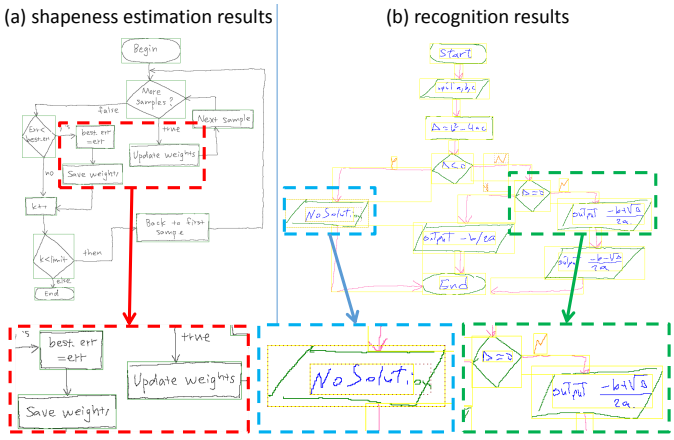

Figure 6: Example results of the shapeness estimation (a) and sketch parsing (b), using stroke level criteria. In (a), green bounding box represents a detected shape. In (b), yellow bounding box represents a recognized symbol, within which different colors of strokes mean different labels. Black dotted box represents a groundtruth symbol that we failed to recognize. These results show that our method can accurately separate the shape edges from noisy sketch of flowchart (e.g., red and green slashed boxes), but might fail in some challenging cases where some text stroke overlap with the shapes (e.g., blue slashed box).

For bounding box criteria, BING achieved 95.0% recall using 1,000 proposals, while our method reached 96.2% recall using only 32 proposals. It also verified the necessary of the shapeness estimation to reduce the number of proposals.

4.2 Sketch Segmentation and Recognition

We follow the same settings and compare with the results in [Lemaitre *et al.*, 2013]. For this evaluation, we conducted two tasks: 1) the labeling of each stroke, and 2) the segmentation and recognition of the symbols. A stroke is correctly labeled if the result label is consistent with the ground truth. A symbol is correctly segmented and recognized if the strokes corresponding to the symbol are exactly the same as that in the ground truth, and that the label of the symbol is correct.

The comparison results are shown in Table 1 and 2. We can see the superiority of the proposed method over compared algorithms: [Bresler *et al.*, 2013], [Lemaitre *et al.*, 2013], and [Carton *et al.*, 2013]. The accuracy was not very high because we used the stroke level criteria which is quite strict. For example, as shown in the blue box of Fig. 6(b), some text strokes overlap with the parallelogram, and thus lead to wrong segmentation. However, this case will be considered as correct in the bounding box criteria. The time cost of our method is 0.54s, mostly owning to the speedup of shapeness estimation. Note that our method did not leverage the temporal information of strokes, while all compared methods did. It can be inferred that in the face of offline sketch parsing without temporal information, other methods will cost more time to ensure a similar recognition rate. To verify the effectiveness of the shapeness estimation in the whole framework, we also implemented the version without the shapeness estima-

tion module, denoted by Ours(without shapeness). We can see that, without shapeness estimation, the accuracy become much worse when removing the shapeness estimation module. This is because the shapeness estimation significantly reduce the false positive candidate groups, making the shape recognition in the next step easier.

Fig. 6 visualized an example of shapeness estimation and sketch parsing with both good and failed results.

5 Conclusion

In this work, we proposed a practical solution for offline sketch parsing, which is more challenging than online sketch parsing due to the unavailability of temporal information. In order to reduce the number of candidate stroke groups in traditional *selection-recognition* approach, we proposed the concept of shapeness, as well as an efficient shapeness estimation algorithm to reduce the number of false positives, resulting in a 96.2% detection rate with only 32 candidate group proposals, which was two orders of magnitude less than counterpart algorithms. Based on the proposed shapeness estimation algorithm, we presented an effective three-stage framework for offline sketch parsing. Experiments on benchmark dataset showed both the effectiveness and efficiency of the proposed framework.

6 Acknowledgements

The work of the first and the third authors was supported by the National Natural Science Foundation of China (Grant Nos 61272251, 91120305) and the National Basic Research Program of China (Grant No. 2015CB856004).

References

[Alexe *et al.*, 2012] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2189–2202, 2012.

[Anderson, 1978] Kenneth R. Anderson. A reevaluation of an efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1978.

[Bresler *et al.*, 2013] Martin Bresler, Daniel Prua, and Václav Hlavác. Modeling flowchart structure recognition as a max-sum problem. In *ICDAR*, 2013.

[Carton *et al.*, 2013] Cérès Carton, Aurélie Lemaitre, and Bertrand Couasnon. Fusion of statistical and structural information for flowchart recognition. In *ICDAR*, 2013.

[Cheng *et al.*, 2014] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *CVPR*, 2014.

[Futrelle and Nikolakis, 1995] Robert P. Futrelle and Nikos Nikolakis. Efficient analysis of complex diagrams using constraint-based parsing. In *ICDAR*, 1995.

[Hammond and Paulson, 2011] Tracy Hammond and Brandon Paulson. Recognizing sketched multistroke primitives. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2011.

[Hse and Richard Newton, 2005] Heloise Hwawen Hse and A Richard Newton. Recognition and beautification of multi-stroke symbols in digital ink. *Computers & Graphics*, 29(4):533–546, 2005.

[Kara and Stahovich, 2004] Levent Burak Kara and Thomas F Stahovich. An image-based trainable symbol recognizer for sketch-based interfaces. In *AAAI Fall Symposium*, 2004.

[Kovari, 2007] Bence Kovari. Time-efficient stroke extraction method for handwritten signatures. In *Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Computer Science - Volume 7*, ACS'07, 2007.

[Lemaitre *et al.*, 2013] 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*. 2013.

[LHomer, 2000] Eric LHomer. Extraction of strokes in handwritten characters. *Pattern Recognition*, 33:1147–1160, 2000.

[Ouyang and Davis, 2009] Tom Y. Ouyang and Randall Davis. A visual approach to sketched symbol recognition. In *IJCAI*, 2009.

[Paulson and Hammond, 2008] Brandon Paulson and Tracy Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, 2008.

[Sezgin and Davis, 2008] Tevfik Metin Sezgin and Randall Davis. Sketch recognition in interspersed drawings using time-based graphical models. *Computers & Graphics*, 32(5):500–510, 2008.

[Shilman and Viola, 2004] Michael Shilman and Paul Viola. Spatial recognition and grouping of text and graphics. In *Proceedings of the First Eurographics Conference on Sketch-Based Interfaces and Modeling*, 2004.

[Sun *et al.*, 2012a] Zhenbang Sun, Changhu Wang, Liqing Zhang, and Lei Zhang. Free hand-drawn sketch segmentation. In *ECCV*. 2012.

[Sun *et al.*, 2012b] Zhenbang Sun, Changhu Wang, Liqing Zhang, and Lei Zhang. Query-adaptive shape topic mining for hand-drawn sketch recognition. In *Proceedings of the 20th ACM International Conference on Multimedia*, 2012.

[Uijlings *et al.*, 2013] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[Wu *et al.*, 2014] Jie Wu, Changhu Wang, Liqing Zhang, and Yong Rui. Sketch recognition with natural correction and editing. In *AAAI*, 2014.

[Yu *et al.*, 1997] Yuhong Yu, A. Samal, and S.C. Seth. A system for recognizing a large class of engineering drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):868 – 890, 1997.