Segmenting Handwritten Math Symbols Using AdaBoost and Multi-Scale Shape Context Features

Lei Hu
Department of Computer Science
Rochester Institute of Technology, USA
lei.hu@rit.edu

Richard Zanibbi
Department of Computer Science
Rochester Institute of Technology, USA
rlaz@cs.rit.edu

***Abstract*—This paper presents a new symbol segmentation method based on AdaBoost with confidence weighted predictions for online handwritten mathematical expressions. The handwritten mathematical expression is preprocessed and rendered to an image. Then for each stroke, we compute three kinds of shape context features (stroke pair, local neighborhood and global shape contexts) with different scales, 21 stroke pair geometric features and symbol classification scores for the current stroke and stroke pair. The stroke pair shape context features covers the current stroke and the following stroke in time series. The local neighborhood shape context features includes the current stroke and its three nearest neighbor strokes in distance while the global shape context features covers the expression. Principal component analysis (PCA) is used for dimensionality reduction. We use AdaBoost with confidence weighted predictions for classification. The method does not use any language model. To our best knowledge, there is no previous work which uses shape context features for symbol segmentation. Experiment results show the new symbol segmentation method achieves good recall and precision on the CROHME 2012 dataset.**

I. Introduction

Recognition of math expressions includes three major parts: symbol segmentation, symbol recognition and structural analysis [1], [2]. The input data for online handwritten expressions is a set of strokes, and a mathematical symbol may contain more than one stroke. Symbol segmentation aims to transform strokes into a set of symbols [3]. It is the basis of symbol classification and structural analysis, therefore the quality of symbol segmentation determines the quality of the whole math expression recognition system.

II. Related Work

A number of approaches have been proposed for mathematical symbol segmentation. One group of methods are based on X-Y cut. Faure and Wang [4] propose a segmentation method based on a modular system which contains two modules: data-driven segmentation module and knowledge-driven segmentation module. Okamoto et al. [5] present a segmentation method based on recursive projection profile cutting. This segmentation method leads to over segmentation. Then the over-segmented symbols are combined based on some rules and segment connectivity matrix. Ha et al. [6] propose a recursive X-Y cut segmentation method for printed expressions based on a top-down process and a bottom-up (merging) process.

X-Y cut or projection methods work well for printed expressions but not for handwritten ones, because handwritten expressions have more variation, and it is hard to segment them well just based on gaps along horizontal or vertical directions.

There have been many graph-based methods [7]–[10]. Those segmentation methods are usually a part of corresponding math expression recognition systems which try to optimize segmentation, symbol classification and structural analysis simultaneously. In those systems, a set of symbol hypotheses will be generated and each hypothesis has a score produced by the segmenter, classifier or structural analyzer. Those symbol hypotheses will form a graph, in which each node represents a symbol candidate and each edge denotes the relationship between two symbol candidates. The segmentation is acquired when the best symbol candidate set is chosen to optimize the evaluation score. So the time complexity is exponential in the number of strokes, all graph based segmentation methods have some constraints to reduce computation: only successive strokes in time series can belong to the same symbol and a symbol at most has 4 strokes [7]–[9], or only connected subtrees in the Minimum Spanning Tree (MST) over strokes can form the symbol candidates [10].

Winkler et al. [7] propose a symbol segmentation method for online handwritten expressions based on a symbol hypotheses net (SHN). Each path in the SHN represents a symbol hypothesis sequence. Stroke specific features are used to classify each stroke to three complexity categories: primitive, standard and complex. Rules specify that only certain combinations of these categories are possible. The segmentation score for each symbol hypothesis is computed based on the stroke specific features and geometric features between strokes. Prerecognition is used to detect dot, minus and fraction lines, and these classification results are used to improve the segmentation rate using rules for these specific symbols before segmenting the other symbols.

Toyozumi et al. [8] present a symbol segmentation method for handwritten expressions based on candidate character lattice method. Evaluation values are calculated based on the positional relation of two strokes and math structure information. Nearest point distance between two strokes is used to get the evaluation value for stroke positional relation. Spatial grammars are defined on structure symbols (such as $-, \sqrt{}, \sum$ and so on) and stroke patterns which satisfy the grammars will be counted up to get the probability of existence of structure symbols. Feedback from symbol classifier is used to improve the segmentation rate.

Shi et al. [9] present a unified probabilistic framework for symbol segmentation and recognition of handwritten expressions based on a symbol graph. They define a function to calculate the grouping likelihood between two strokes by using three geometric features (horizontal distance, size difference and vertical offsite).

Matsakis [10] proposes a segmentation method based on MST. For the given expression, it will form an MST over the strokes. In the MST each node represents a stroke, and the distance between any two strokes is the Euclidean distance between the centers of the bounding boxes. The segmentation method only considers partitions that form connected subtrees in the MST.

Another group of methods [11]–[15] are driven by grammars. There are some segmentation methods [11]–[13] under the restriction of a set of math expression grammars. For these segmentation methods, a symbol candidate generator produces multiple symbol candidates. A cost function is defined to choose the final interpretation. The grammars are used to check the validity of proposed interpretations. In [11], the two dimensional math expression grammars are combined by two sets of one dimensional grammars on horizontal and vertical direction. Maximum number of hypothesis, maximum number of strokes per symbol and maximum distances between strokes forming the same hypothesis are limited. In [12], a set of LL(1) grammars is defined and maximum number of strokes per symbol is limited. In [13], there is a set of fuzzy relational context free grammars.

Some segmentation methods [14], [15] are driven by stochastic context free grammars (SCFG). In [14], the expressions recognition system is driven by the 2D context-free probabilistic graph grammar to find all mathematically valid interpretations and assign scores to each possible interpretation. The suitability of the symbols spatial distribution for the rules and the likelihoods of the recognized symbols determine the score of an interpretation. In [15], the online handwritten expressions will be rendered to image first. Then each connected component will be taken as a symbol candidate and passed to the symbol classifier based on Hidden Markov Model (HMM). The SCFG determines whether to merge two symbol candidates or not based on the class label and class confidence produced by the symbol classifier.

There have also been many other methods [16], [17]. Smithies et al. [16] present a progressive segmentation method. It has the time sequential assumption. For 4 strokes, the segmenter generates all possible groupings. Then the first recognized character from the group with highest confidence level will be removed and the process will restart when there is 4 strokes again. Kosmala et al. [17] propose a segmentation method based on HMM. Discrete left to right HMMs without skips and with different numbers of states are used. An addition space model is introduced to model the spaces between symbols.

III. Methodology

Like many previous segmentation methods, our method assumes a symbol can only contain successive strokes. But our method does not specify the number of strokes a symbol can have and does not use any language model. Given an expression with n strokes, our segmentation method just considers merging or splitting the $n-1$ stroke pairs $(S_1, S_2), (S_2, S_3), ..., (S_{n-1}, S_n)$ in time sequence and only provides one segmentation interpretation. The time complexity of our segmentation method is $O(N^2)$. Therefore our segmentation method is efficient, in terms of computation.

Expressions are preprocessed to reduce noise and resolution variance first. Then for each stroke pair, we extract 351 features, including 21 geometric features, three types of shape context features (each shape context feature has 60 dimensions) and two sets of classification scores (each set of classification scores has 75 dimensions). After the feature extraction, we apply PCA to the 351 features and choose the first 100 components. The 100 components are used to train the classifier by using AdaBoost with confidence weighted predictions [18].

A. Preprocessing

To reduce noise and resolution variance between different expressions, we apply preprocessing to render the expression to an image. Our procedure contains four steps: duplicate point filtering, smoothing, size normalization and resampling.

We delete duplicate point which has the same (x,y) coordinates as the previous point because they cannot provide any useful information. To reduce the noise caused by the stylus' jogging, we smooth the whole expression. Except the first point and the last point of each stroke, we replace the other points' coordinates by the average of the coordinates of current point, the previous point and the next point. In order to remove the influence of writing velocity and devices' difference in coordinate range and resolution, we transform the y coordinate's range to be [0, 200] while preserving the width-height aspect ratio. Then we use linear interpolation to resample the expression and render it to an image.

B. Feature Extraction

All the stroke pair features used in previous segmentation methods are geometric features. Winkler et al. [7] use minimum distance, horizontal overlapping of the bounding box, distance and offset between the beginning points and end points, backward movement and parallelity of two successive strokes. Shi et al. [9] use horizontal distance, size difference and vertical offsite. Toyozumi et al. [8] use the minimum point distance. MacLean et al. [13] use the overlapped area.

We use all the geometric features mentioned above and design some other geometric features: distance between bounding box centers, distance between averaged centers (the coordinates of averaged center are the average of the coordinates of all points of the stroke), maximal point pair distance (two points are from different strokes of the stroke pair), horizontal offset between the ending point of the first stroke and the beginning point of the second stroke, vertical distance between bounding box centers, writing slope (slope is the angle between the horizontal line and the line connecting the last point of the current stroke and the first point of the next stroke) and writing curvature (curvature is angle between the line connecting the first point and last point of the current stroke and the line connecting the first point and last point of the next stroke). We normalize all the geometric features except parallelity, writing slope and writing curvature to make them between [0, 1].

It is hard to design a good set of geometric features, so we add multi-scale shape context features (MSSCF). The shape context [19] at a given point captures the distribution of the other points relative to it. Figure 1 shows one example of shape context feature. A circle is divided into 60 bins, containing 12 equal angle bins and 5 distance bins. We chose the parameters for bins based on Belongie et. al's paper [19]. The ratios of the radii of the five distance bins to radius of the outmost circle are $\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}$ and 1. For each bin, the number of points within that bin divided by the number of points is a shape context feature.

Ouyang and Zanibbi [20], [21] presented a symbol layout classification method to classify printed math symbols into seven layout classes [22]: ascender, descender, centered, open bracket, non-scripted, variable range and root. Hirata and Honda [23] used shape context feature in their graph matching method to automatically label symbols in handwritten math expressions. Marinai et al. [24] used shape context feature for math symbol retrieval.

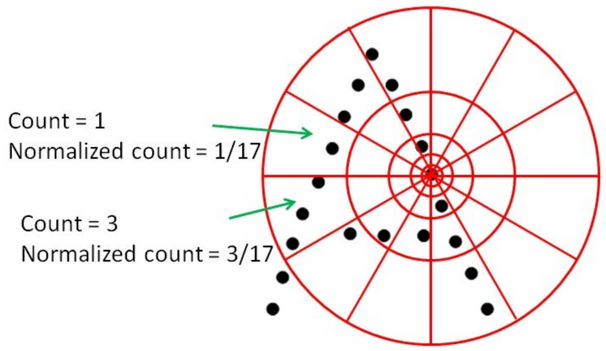

Fig. 1. Example: shape context feature computation.

We design three shape context features: stroke pair, local neighborhood and global. Figure 2 shows one example of multi-scale shape contexts. For the three kinds of shape contexts, the circle center is the center of the bounding box of the current stroke, but their radii will be different. **Stroke pair shape context** only considers the current stroke and the following stroke in time sequence. The radius for stroke pair shape context is the maximal distance between the points in the two strokes and the circle center. **Local neighborhood shape context** considers the current stroke and its three nearest neighbor strokes. The distance between two strokes is the minimal point distance between them. The radius for neighbor shape context is the maximal distance between the points in the four strokes and the circle center. **Global shape context** covers the whole expression. The radius is the maximal distance between the points in the whole expression and the circle center. Each one of the three circles will be divided into 60 bins as showed in Figure 1. The MSSCF capture the successive stroke pair, local neighborhood and the whole expression separately and include information at different level. Compared to a single shape context feature, the MSSCF contains much more information which is helpful for deciding to merge or split the successive stroke pair.

For different symbol classes, the strokes' layout and neighborhoods are different. We add the classification scores as additional features to use the symbol class information. The classifier we use is similar to the HMM classifier in [25], but with additional angular feature and global features [12]. We design a continuous left to right HMM for each symbol class. A variant of segmental K-means is used to get initialization of the Gaussian Mixture Models' parameters which represent the observation probability distribution of the HMMs. We take the current stroke (the first stroke of the stroke pair) as one symbol candidate and also take the stroke pair as the other symbol candidate. For each symbol candidate, the classification scores with a fixed order over all the classes are used as features.

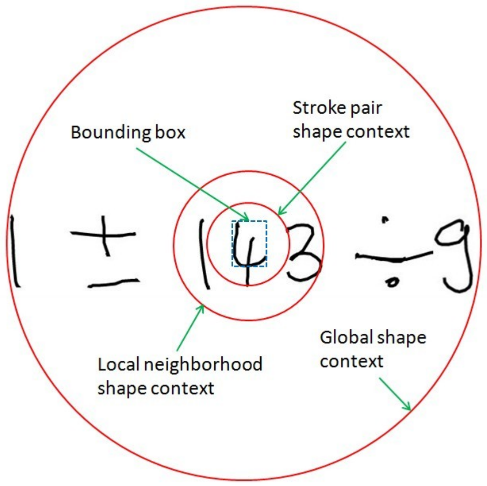

Fig. 2. Example of multi-scale shape contexts (current stroke is the top stroke of the symbol 4).

C. Dimensionality Reduction

For each stroke pair, we have 351 features which are based on the stroke pair, such as geometric features, stroke pair shape context features and classification scores, or are centered around the stroke pair, such as local neighborhood shape context features and global shape context features. Among these features, some are redundant. For example, the bins which are far away from the circle center vertically in global shape context usually contain no points, and those shape context features are always 0. We use PCA to reduce noise and redundancy. After PCA, the first 100 components account for the 99.86% variance of all the features. The ratio of the variance of the first five components are 45.24%, 25.75%, 18.96%, 3.90% and 1.11%. The ratio of the variance of the other components are all less than 1.00% and form a long tail. The shape of the plot about the ratio of the variance of each component in descending order is close to an 'L'. The first 100 components are used as new features to train the classifier.

D. AdaBoost

We use AdaBoost with confidence-rated predictions, and the weak learner is a decision stump. In each iteration of AdaBoost, we try to minimize the normalization factor

$$Z_t \doteq \sum_{i=1}^{m} D_t(i) exp(-\alpha_t y_i h_t(x_i)), \quad (1)$$

where D_t is the weight distribution over the all training samples; α_t is a parameter which is used to update $D_t(i)$; $y_i \in \{-1, +1\}$ ($+1$ represents merge while -1 represents split); h_t is the weak hypothesis.

There are 23424 training samples for our classifier, therefore it is hard to calculate h_t and a_t in general method. By assuming the weak learner can scale any weak hypothesis h by any constant factor $\alpha \in R$ freely without loss of generality, expression (1) can be simplified by folding a_t into h_t [18]. We choose α to be 1. Then our goal becomes minimizing

$$Z_t = \sum_{i=1}^{m} D_t(i) exp(-y_i h_t(x_i)). \tag{2}$$

Our classifier only contains two classes: merge and split. The weak learner decision stump finds a threshold to divide the whole training samples into two parts. In each iteration, for all samples within each block X_j, weak hypothesis h is equal to some fixed value c_j. The normalization factor in equation (2) is minimized when

$$c_j = \frac{1}{2} ln \left(\frac{W^j_{\text{merge}} + \epsilon}{W^j_{\text{split}} + \epsilon} \right), \tag{3}$$

where W^j_{merge} is the summed weight of merge samples which fall in block j while W^j_{split} is the summed weight of split samples which fall in block j. ϵ in expression (3) is typically chosen to be on the order of $\frac{1}{m}$ and m is the number of training samples [18]. We choose ϵ to be $\frac{1}{23424}$.

IV. Dataset and experiment results

The dataset we use in this paper is the Part-III dataset in CROHME 2012 [26]. We use the training set of Part III data to train the classifier and the testing set for test. The training set has 1338 expressions and 23424 successive stroke pairs, in which 7773 stroke pairs should be merged while 15651 should be split. The testing set has 488 expressions and 8379 successive stroke pairs, in which 2459 stroke pairs should be merged while 5920 should be split. Training set and testing set have the same 75 symbol classes.

We train the classifier 10000 iterations, and Table I shows the minimal classification error rates on the successive stroke pairs of training set and testing set with different features among the 10000 rounds.

TABLE I. Minimal error rates (corresponding numbers of iteration) on the successive stroke pairs of training set and testing set with different features among the 10000 rounds (G: geometric features, MSSCF: multi-scale shape context features, SPSCF: stroke pair shape context features, LNSCF: local neighborhood shape context features, GSCF: global shape context features, C: classification scores).

features	training set		testing set	
G	7.60%	(9995)	12.85%	(390)
MSSCF	1.50%	(9999)	14.49%	(822)
SPSCF	5.72%	(9995)	15.85%	(5366)
LNSCF	14.48%	(9909)	25.27%	(542)
GSCF	15.48%	(9965)	29.26%	(12)
G + MSSCF	0.00%	(9946)	10.36%	(6604)
G + MSSCF + C	0.00%	(7147)	8.80%	(4111)

From Table I, we can find that MSSCF performs better than G (geometric features) on training set but worse on testing set. This shows MSSCF is easy to overfit. MSSCF performs much better than either one of three shape context features (SPSCF, LNSCF and GSCF) on both training and testing set. If we just use one of the three shape context features, the larger the radius is, the worse the performance. The reason is that the shape context features will have less variance and become less discriminative when the scope become larger. Using G and MSSCF can get better performance on training set and testing set than using either one of them alone. Adding classification scores to G and MSSCF can improve the performance further.

For the classification error rate curve on training set and testing set over the 10000 iterations by using G + MSSCF + C, the error rates on training set and testing set drop fast in the first 2000 iterations. After 2000 iterations, the training error rates continue to drop and are almost 0 after 6000 iterations, while the testing error rates just change a little. The error rate curves by using the other sets of features (such as G or MSSCF or G+MSSCF) have the similar shape. This shows that AdaBoost with confidence-rated predictions will not overfit the training set.

Figure 3 shows the precision and recall on testing set with different features. Recall is the same as symbol segmentation rate in [26]. We can find that the precision and recall for different features on training set or testing set is very close. It is easy to find that using G + MSSCF + C gets the best performance. By using all these three sets of features, our segmentation method achieves 99.88% recall, 99.76% precision on training data and 84.95% recall, 84.79% precision on testing data. In CROHME 2012 [26], the top two systems (MyScript Equation recognizer from Vision Objects and the Waterloo recognizer [13]) get 98.84% and 95.56% symbol segmentation rate (recall) on the testing data. The reasons there is a gap between our segmentation method and the top two systems are: (1) given an expression, these two systems produce multiple segmentations while our segmentation method produces one; (2) both systems use language models while our segmentation method does not; (3) both systems use extra datasets for training; (4) both segmentation methods cannot be separated from the whole math expression recognition systems which optimize the segmentation, classification and parsing at the same time while our segmentation method is independent.

Through analyzing the experiment results, we find many segmentation errors happen to multi-stroke symbols, such as cos, sin, log, lim, ellipsis ... and so on. Those multi-stroke symbols are easy to be over segmented. Symbol segmentation rates for cos, sin, log, lim, ellipsis ... are 38.37%, 48.00%, 44.44%, 34.69% and 0. For the single-stroke symbols, which have intersections with nearby symbols or are very close to nearby symbols, they are easy to be under-segmented. Therefore the symbol segmentation rate can be improved by adding prerecognition. Prerecognition means trying to find symbol candidate which is highly possible to be a symbol and this symbol is not a part of other symbol. The strokes of the symbol candidate will be isolated from the other strokes.

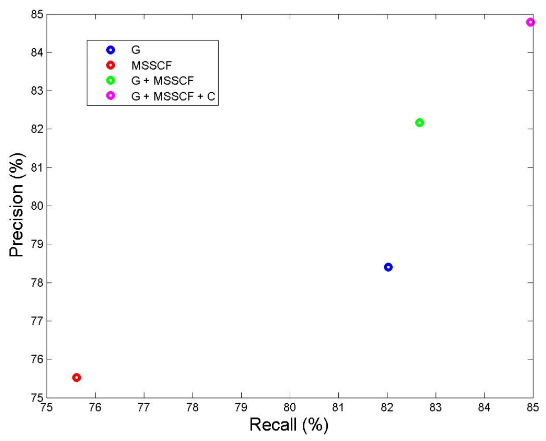

Fig. 3. Precision and recall on testing set with different features (G: geometric features, MSSCF: multi-scale shape context features, C: classification scores).

V. Conclusion

This paper presents a new symbol segmentation method for online handwritten mathematical expressions using AdaBoost with confidence rated predictions. For the stroke pair containing two successive strokes in time series, we calculate geometric features, a new shape context-based feature (multi-scale shape context features) and classification scores. Multi-scale shape context features includes three different shape contexts: stroke pair, local neighborhood and global shape contexts. Then we apply PCA to reduce dimensionality. AdaBoost with confidence rated predictions is the classifier. Our segmentation method achieves 84.95% recall, 84.79% precision on testing data. The initial experiment results are encouraging in light of we do not use any language model.

In future work, we will make symbol segmentation interact more with symbol classification and structural analysis to get higher recall.

Acknowledgment

This material is based upon work supported by the National Science foundation under Grant No. IIS-1016815. The authors would like to thank David Stalnaker and Francisco Álvaro for valuable discussions.

References

[1] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *International Journal on Document Analysis and Recognition*, vol. 15, no. 4, pp. 331–357, 2012.

[2] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition*, vol. 3, no. 1, pp. 3–15, Aug. 2000.

[3] R. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 18, no. 7, pp. 690 –706, jul 1996.

[4] C. Faure and Z. Wang, "Automatic perception of the structure of handwritten mathematical expressions," *Computer Processing of Handwriting*, pp. 337 –361, 1990.

[5] N. Okamoto and B. Miao, "Recognition of mathematical expressions by using the layout structures of symbols," in *Proc. Int'l Conf. on Document Analysis and Recognition*, Saint-Malo, France, 1991, pp. 242–250.

[6] J. Ha, R. Haralick, and I. Phillips, "Understanding mathematical expressions from document images," in *Proc. Int'l Conf. on Document Analysis and Recognition*, Aug 1995, pp. 956–959.

[7] S. Lehmberg, H.-J. Winkler, and M. Lang, "A soft-decision approach for symbol segmentation within handwritten mathematical expressions," in *Proc. Int'l Conf. on Acoustics, Speech, and Signal Processing*, May 1996, pp. 3434–3437.

[8] K. Toyozumi, N. Yamada, T. Kitasaka, K. Mori, Y. Suenaga, K. Mase, and T. Takahashi, "A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information," in *Proc. Int'l Conf. on Pattern Recognition*, Aug. 2004, pp. 630–633.

[9] Y. Shi, H. Li, and F. Soong, "A unified framework for symbol segmentation and recognition of handwritten mathematical expressions," in *Proc. Int'l Conf. on Document Analysis and Recognition*, Sept. 2007, pp. 854 –858.

[10] N. Matsakis, "Recognition of handwritten mathematical expressions," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, May 1999.

[11] A.-M. Awal, H. Mouchre, and C. Viard-Gaudin, "A global learning approach for an online handwritten mathematical expression recognition system," *Pattern Recognition Letters*, 2012.

[12] L. Hu, K. Hart, R. Pospesel, and R. Zanibbi, "Baseline extraction-driven parsing of handwritten mathematical expressions," in *Proc. Int'l Conf. on Pattern Recognition*, Nov. 2012, pp. 326 –330.

[13] S. MacLean and G. Labahn, "A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets," *International Journal on Document Analysis and Recognition*, pp. 1–25, 2012.

[14] M. Celik and B. Yanikoglu, "Probabilistic mathematical formula recognition using a 2d context-free graph grammar," in *Proc. Int'l Conf. on Document Analysis and Recognition*, Sept. 2011, pp. 161 –166.

[15] F. Álvaro, Sánchez, and J. Benedí, "Recognition of on-line handwritten mathematical expressions using 2D stochastic context-free grammars and hidden Markov models," *Pattern Recognition Letters*, 2012.

[16] S. Smithies, K. Novins, and J. Arvo, "A handwriting-based equation editor," *Proc. Graphics Interface*, pp. 84–91, June 1999.

[17] A. Kosmala and G. Rigoll, "On-line handwritten formula recognition using statistical methods," in *Proc. Int'l Conf. on Pattern Recognition*, 1998, pp. 1306–1308.

[18] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[19] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509 –522, Apr 2002.

[20] L. Ouyang, "A Symbol layout classification for mathematical formula using layout context," Master's thesis, Rochester Institute of Technology, Rochester, NY, 2009.

[21] L. Ouyang and R. Zanibbi, "Identifying layout classes for mathematical symbols using layout context," in *Proc. IEEE Western New York Image Processing Workshop*, 2009.

[22] R. Zanibbi, D. Blostein, and J. R. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455–1467, Nov. 2002.

[23] N. S. T. Hirata and W. Y. Honda, "Automatic labeling of handwritten mathematical symbols via expression matching," in *Proc. Int'l Conf. on Graph-based Representations in Pattern Recognition*, 2011, pp. 295–304.

[24] S. Marinai, B. Miotti, and G. Soda, "Using earth mover's distance in the bag-of-visual-words model for mathematical symbol retrieval," in *Proc. Int'l Conf. on Document Analysis and Recognition*, Sept. 2011, pp. 1309 –1313.

[25] L. Hu and R. Zanibbi, "HMM-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature," in *Proc. Int'l Conf. on Document Analysis and Recognition*, Sept. 2011, pp. 457 –462.

[26] H. Mourchère, C. Viard-Gaudin, D. Kim, J. H. Kim, and U. Garain, "ICFHR 2012 competition on recognition of on-line mathematical expressions (CROHME 2012)," in *Proc. Int'l Conf. on Frontiers in Handwriting Recognition*, Sept. 2012, pp. 811 –816.