A New Approach for Context-Independent Handwritten Offline Diagram Recognition using Support Vector Machines

Khaled S. Refaat, Wael N. Helmy, AbdelRahman H. Ali, Mohamed S. AbdelGhany, Amir F. Atiya

Abstract— **Structured diagrams are very prevalent in many document types. Most people who need to create such diagrams use structured graphics editors such as Microsoft Visio [1]. Structured graphics editors are extremely powerful and expressive but they can be cumbersome to use [2]. We have shown through extensive timing experiments that structured diagrams drawn by hand will take only about 10% of the time it takes to draw one using a tool like Visio. This indicates the value of automated recognition of hand-written diagrams. Recently, applications have been developed that use online systems running on pen-input PCs that allow users to create structured diagrams by drawing the diagram on the PC tablet. The progress of offline diagram recognition is still minimal. The objective of this paper is to propose a context-independent off-line diagram recognition system. Our approach utilizes support vector machines [3] for recognition and Line Primitive Extraction by Interpretation of Line Continuation for segmentation [4].**

I. Introduction

Structured diagrams (flow charts, Markov chains, module dependency diagrams, state diagrams, block diagrams, UML, graph…etc) have become indispensable tools for the business world. They are prevalent in all types of documents, whether business reports, research publications, white papers, etc. Most people who need to create such diagrams use structured graphics editors such as Microsoft Visio [1]. Structured graphics editors are extremely powerful and expressive but they can be cumbersome to use. We have performed extensive experiments comparing the times of hand-drawn structured diagrams with those entered using a tool like Visio. The result is that the hand-drawn diagrams take on average 90% less time. This suggests that an automated solution, based on hand-written diagram recognition, will save a significant amount of time for the user, and hence will be economically very beneficial. Recently, applications have been developed that use online systems running on pen-input PCs that allow users to create diagrams by drawing on the PC tablet. These systems use the so-called online diagram-recognition approach. The other harder type is the off-line diagram recognition approach, which is based on processing the image as a whole after it has been completely drawn. The online approach makes use of the sequence of strokes, and this is a very facilitating piece of information. While there have been several works in the literature on online recognition, the progress of offline diagram recognition is still very minimal. This is partly due to the difficulty of the problem. In this paper we propose a system of off-line diagram recognition. The proposed model consists of all possible components of a diagram recognition system, such as segmentation, feature extraction, classification, and redrawing and repositioning. For the first component, we need to segment the diagram into standalone shapes. For that, our approach utilizes Line Primitive Extraction by Interpretation of Line Continuation [4]. We propose a novel adjustment to this algorithm in which we enhance the output of segmentation by reattaching shapes that have been erroneously divided. In addition, we propose a novel algorithm that detects hand-drawn lines and distinguishes them from shapes. In the second component size and orientation independent features should be extracted from each shape. We propose a group of effective features that characterize the different shapes. The third component is the classifier, which is for the purpose of recognizing the shapes based on the extracted features. Here we use support vector machines (SVM) [3] to classify the shapes. Finally, in the last component, shapes are redrawn in the same relative position and with the same drawn size to produce a professional diagram similar to that produced by structured graphics editors currently on the market.

This paper is organized as follows: Section (II) describes the related work. Segmentation will be presented in Section (III). Section (IV) describes the Feature extraction stage. The Support Vector Machine as a multi-class classifier for recognition of shapes is presented in section (V). Finally we introduce the experimental results in section (VI). The paper ends with a conclusion and future work in section (VII).

II. Related work

Previous research in hand-drawn diagram understanding has generally followed one of two paths: Online interfaces that require an instrumented drawing surface that captures stroke information or off-line interfaces that allow the user to sketch using pen-and-paper. Lank *et al* [5] used online stroke information to recognize UML diagrams. UML symbols were recognized by heuristics such as stroke count and the ratio of stroke length to bounding-box perimeter. Its user interface allows the user to correct the system at each stage of understanding: acquisition, segmentation, and recognition. Tahuti [6] is another online system for UML diagram recognition, which uses both geometric and contextual constraints to recognize UML symbols. The system is non-modal: users can interleave editing and

The authors are with the Computer Engineering Department, Faculty of Engineering, Cairo University, Egypt (e-mails:khaled.saeed84@gmail.com, alaa.wael@gmail.com,abdelrahman.hasan@gmail.com, mr.sarwat@gmail.com , and amir@alumni.caltech.edu).

978-1-4244-1821-3/08/$25.00©2008 IEEE

drawing with no mode switching, which requires editing gestures to be distinguishable from sketched symbols. Denim [7] is an online system for informal sketching of web site designs, in which the user can sketch boxes to represent web pages and lines to indicate links between the pages. Different kinds of links are recognized based on their context rather than their shape. The system uses a unistroke recognizer [8], so each symbol must be sketched with a single stroke. Sezgin *et al* [9] have created an online system that recognizes geometric primitives. The system uses a three-phase technique (approximation, beautification, basic recognition) to transform sketches into sharpened diagrams. Its recognition process uses shape and curvature of strokes, among other information. One of the earliest tools in the online category was SILK [10], which recognized sketches of user interfaces. The sketch could then be converted into an interactive demo of the sketched GUI. Jorge and Fonseca presented a novel approach to recognizing geometric shapes interactively [11], using fuzzy logic and decision trees to recognize multi-stroke sketches of geometric shapes. Hse and Newton presented an online recognition method for hand-sketched symbols [12]. This method is independent of stroke-order, number, and direction, as well as invariant to rotation, scaling, and translation of symbols. Zernike moment descriptors are used to represent symbols and three different classification techniques are compared: support vector machines (SVM), minimum mean distance (MMD), and nearest neighbor (NN).

Research on offline diagram interpretation, using scanned images of pen-and-paper diagrams is, however, less common. Valveny and Marti discuss a method for recognizing handdrawn architectural symbols [13] using deformable template matching. They achieve recognition rates around 85%, but do not discuss how the user might correct an incorrect recognition. Ramel [14] presents a technique for recognizing handwritten chemical formulas, with a text localization module that extracts text zones for an external OCR system. They also use a global perception phase that incrementally extracts graphical entities. Their system achieves recognition rates of 95-97%, but as in Valveny and Marti, there is no discussion of how the user can correct recognition errors. Notowidigo and Miller [15] presented a novel approach to creating structured diagrams. There system aims to provide drawing freedom by allowing the user to sketch entirely off-line using a pure pen-and-paper interface. The system can infer multiple interpretations for a given sketch to aid during the user's polishing stage. The UDSI program uses a novel recognition architecture that combines low-level recognizers with domain-specific heuristic filters and a greedy algorithm that eliminates incompatible interpretations.

III. SEGMENTATION

Before introducing the image to the SVM classifier a first step is to extract the different shapes that constitute the image. The foreground of the image in our problem consists of text and graphics. The graphics part is made of a set of line primitives, therefore segmentation needs to be done in three steps:

1. Extracting the foreground from the background.
2. Separating text from graphics.
3. Extracting line primitives that construct the graphics in the foreground.

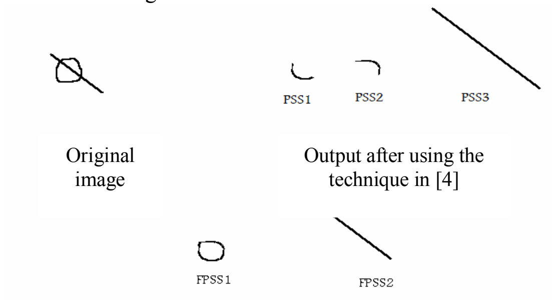
Adaptive thresholding method [16] is chosen for the first step. In the second step, it is easy to see that the text components have a relatively small size if compared to the graphics' components, so we can apply a size filter to separate text and graphics components. The line primitive extraction part is the most challenging part, as it has a direct effect on the quality of the classification module. The difficulty of this module originates from the absence of color difference between various lines primitives. Mainly two types of techniques have been invented for solving this problem. The first type depends on sophisticated mathematical models based on probabilistically modeling the interacting primitives [17]. The other type depends on psychological models that try to extract primitives by studying the way by which humans do this job and then tries to model it mathematically. We chose to use the technique described in [4] which falls in the second category (i.e. a psychologically based method). This technique mathematically models two of the shape properties used by humans to extract the shape from the diagram. These are the smoothness and the continuity. This technique produces acceptable results. We will call the output of this stage "post segmentation shapes" (PSSs).

Segmentation Error Fixer:

After applying the technique in [4], the resultant output may contain some errors like dividing one intrinsic shape into parts. To enhance the output of the segmentation module, another module was added after it to reattach the shape parts which were separated by fault. This added module is called the "segmentation error fixer".

The segmentation error fixer module makes use of the fact that erroneously segmented shape will often produce subparts having near endpoints. The proposed module builds a table indicating the shapes that should be attached. This is basically a map of the attachment relations between all the PSSs. The attachment between two PSSs is detected if one of them has an endpoint that is very close to an endpoint of the other. An endpoint is considered near to another if the distance between them was smaller than a pre-specified threshold that could be experimentally pre-determined. PSSs that were marked to be attached into one object are combined to give only one PSS. The outputs of this stage are called Fixed PSSs or FPSSs. Figure 1 shows an example of an erroneous segmentation result that is subsequently fixed by the segmentation error fixer. As shown in figure1, PSS1 and PSS2 were separated in the segmentation stage, although they perceptually seem to be one logical object, namely a circle. We build a table to indicate the shapes that should be attached as shown in table1. It was found that PSS1 has two endpoints each of them is close to another corresponding endpoint in PSS2 according to the pre-specified threshold.

Consequently PSS1 and PSS2 are marked to be attached in the table, after applying the fixing algorithm PSS1 and PSS2 are combined together.



Output after applying the segmentation error fixer

Fig. 1. an example of an erroneous segmentation that has been fixed by the segmentation error fixer

TABLE I
TABLE OF SHAPE ATTACHMENTS FOR THE EXAMPLE IN FIG. 1.

	PSS 1	PSS 2	PSS 3
PSS 1		Attached	
PSS 2			
PSS 3			

Line detection algorithm:

Hand-drawn lines need a very complex set of features to be classified correctly as they may take quite irregular paths between different objects, as shown in figure 2.

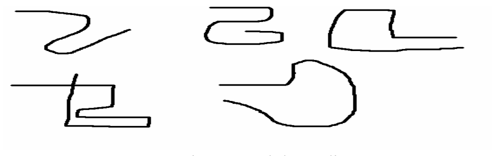


Fig. 2. Hand-drawn lines

To tackle this situation, we have introduced a module between the segmentation stage and the classification stage to classify lines with acceptable accuracy without the need to apply the classifier.

Before we start the illustration of our algorithm we first define a "line" (or a connector) as a connection between FPSSs (fixed post segmentation shapes) and it does not contain a hole. We also check the connectivity of an endpoint to a FPSS by examining the $\beta \times \beta$ square of neighbors including the endpoint itself, where β is determined practically as it depends on the resolution. If any of the neighbors are connected to the FPSS then we interpret that the endpoint is connected to such FPSS as shown in figure 3.

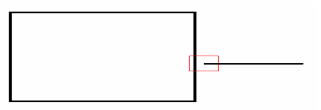


Fig. 3. The end point of the line is considered connected to the FPSS, the smaller rectangle is the $\beta \times \beta$ neighborhood of the endpoint of the line)

The following steps show the proposed algorithm.

Step1: Get the end points of the FPSS by thinning [16] the FPSSs then extracting the points that have only one neighbor in the eight-neighborhood [16].

Step2: After extracting the endpoints of each FPSS we have one of the following *cases:*

1-The FPSS has no end-points: In this case it will be interpreted as a shape (i.e. not a line).

2-The FPSS has one or more end-points and none of these end-points is connected to another FPSS: Also in this case the FPSS is interpreted as a shape. Figure 4 shows a shape that has an end point that is not connected to another FPSS.

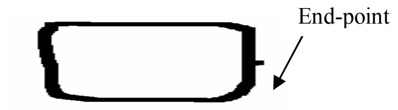


Fig. 4. A FPSS that contains an end-point that is not connected to another FPSS

3-The FPSS has one or more end-points and at least one of these end-points is connected to another FPSS: In this case go to step3.

Step3: In this step we check if the object under consideration has a hole. If it does, then it is interpreted as a shape, otherwise the object is interpreted as a line, as it satisfies the constraints introduced by our previous definition (it has no holes and it is connected to at least one FPSS).

Step4: By now we have two groups of FPSSs, one containing shapes and the other containing lines. The one containing shapes will be introduced to the feature extraction and recognition modules. They will be classified as any geometric shape including the line. The other group, the FPSSs that are lines, is passed directly to the drawing module (no classification needed).

IV. FEATURE EXTRACTION

A number of features are extracted from each shape for the purpose of serving as inputs to the classifier. The features were selected to be size and orientation independent. The basic shape features used are Ach ,the area of the convex hull; Pch, its perimeter; Aer the area of the rectangle enclosing the convex hull of the shape and having the minimum area; Alq, the area of the maximum area inscribed rectangle that fits inside the convex hull of the shape [19]; Alt , the area of the maximum area inscribed triangle that fits inside the convex hull of the shape [19]; Plt, its perimeter [19]; Her, the height of the rectangle enclosing the convex hull of the shape and having the minimum area; and , Wer its width.

The features used are:

- Pch^2 / Ach (Thinness ratio), this feature distinguishes in particular circles from other shapes. The thinness ratio of a circle is minimal, since it is the planar figure with the smallest perimeter enclosing a given area [18].
- Ach / Aer, this feature is useful for distinguishing rectangle shapes from other shapes. The Ach / Aer ratio will have values near unity for rectangles [18].
- Alq / Aer, this feature is good for distinguishing rectangles from other shapes. Alq / Aer ratio will have values near unity for this shape [18].
- Alq / Ach, this feature helps distinguishing ellipses from other shapes. Alq / Ach ratio will have values near 0.7 for ellipses and bigger values for other shapes [18].
- Alt / Alq. This feature distinguishes diamonds from other shapes. The Alt / Alq ratio will have values between 0.5 and 0.6 for diamonds and bigger ones for other shapes [18].
- Alt / Ach, This feature helps distinguishing triangles from other shapes. Alt / Ach will have values near unity for triangles and smaller values for other shapes [18].
- Plt / Pch, this feature distinguishes triangles from other shapes [18]. Plt / Pch will have values near unity for triangles and smaller values for other shapes.
- Her / Wer , this feature can distinguish lines from other shapes [18].

V. SVM CLASSIFIER

For the classification stage we used the SVM classifier. SVM was originally developed for the two class case, but subsequently was generalized to the multi-class case in several studies [20], [21], [22]. From among the proposed multi-class variants we selected to apply the one versus the rest technique [3]. We have used six binary classifiers. Each classifier is associated with a shape (or class). For example, the diamond classifier is concerned with whether the unknown shape is a diamond or not. Each classifier returns a confidence level indicating the confidence in the classification. This confidence level might be negative yielding less confidence or positive yielding more confidence. The classifier with the maximum returned confidence level wins, and the unknown shape is recognized to be of the same type of the winner classifier's associated shape. All binary classifiers use the radial basis function [3] as a kernel with the gamma parameter set to 2.

VI. EXPERIMENTAL RESULTS

After several trial and error experiments with different kernels and parameters, the best results were got using RBF kernel with the gamma parameter set to two. We have used 720 training patterns, 120 for each shape. On the other hand 120 testing patterns were used, 20 for each shape. The β parameter was set to 41. The test classification accuracy turned out to be 90%. The detailed testing accuracies for the different shapes are shown in Table II.

TABLE I I
TESTING RESULTS

	Cir	Tri	Rec	Dia	Ell	Lin	overall
Acc%	100	95	65	95	85	100	90

where Cir, Tri, Rec, Dia, Ell, and Lin stand for circle, triangle, rectangle, diamond, ellipse, and line, respectively, whereas Acc means accuracy. We have not compared this result to other previous works, because we have found no other study covering similar scope and handling similar-type shapes.

It is hard to quantify the performance of the segmentation stage because it is in some way subjective. More than one error could occur in one diagram, and some errors can be much graver than others. It is therefore a good strategy to apply the algorithm to a large number of diagrams, and assess its subjective performance visually. We have therefore created several hand-drawn diagrams, and applied the segmentation and classifier on them. The figures shown in the appendix show the performance of the model (including its segmentation and its classification stages). One can see that the segmentation is performed satisfactorily for all of the tested diagrams.

VII. CONCLUSION AND FUTURE WORK

The field of diagram recognition faces many challenges, including the great diversity in diagrammatic notations, and the presence of noise and ambiguity during the recognition process. In this research work, we used line primitive extraction approach for segmentation and the SVM for classifying the shapes according to their appropriate category. The system performed well in both the segmentation stage and the classification stage. Unfortunately, there is no standardized evaluation for such a real world problem due its great diversity, in our future work we intend to provide a standardized shapes' database after increasing the size of our current one to include more real world shapes.

ACKNOWLEDGMENT

The authors would like to thank *Dr. Nevin Darwish*, chairman of Computer Engineering Department, Cairo University, Egypt for facilitating all resources for this project. The authors would also like to thank *Dr. Osama Emam*, IBM Egypt, for the useful advice he provided.

Appendix

In this appendix we introduce samples of our output.

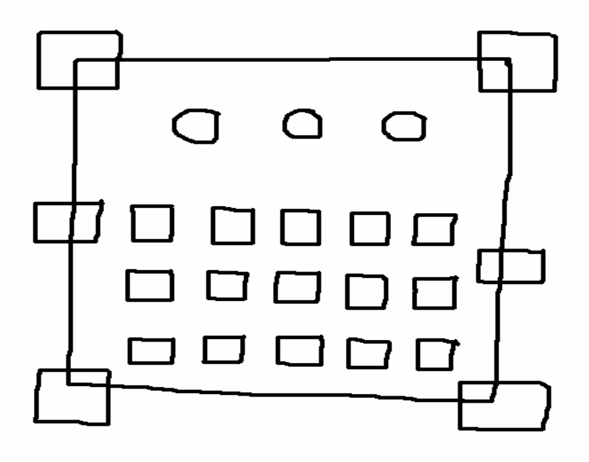


Fig. 5. Input Sample No 1

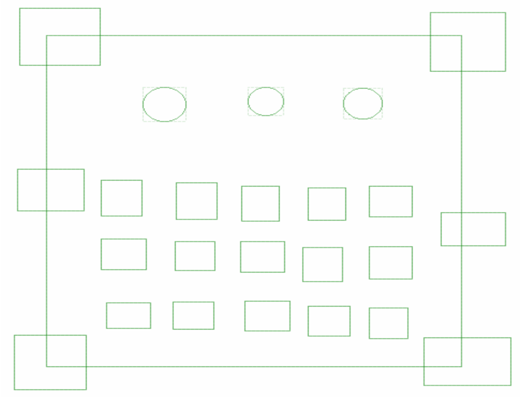


Fig. 6. Output Sample No 1

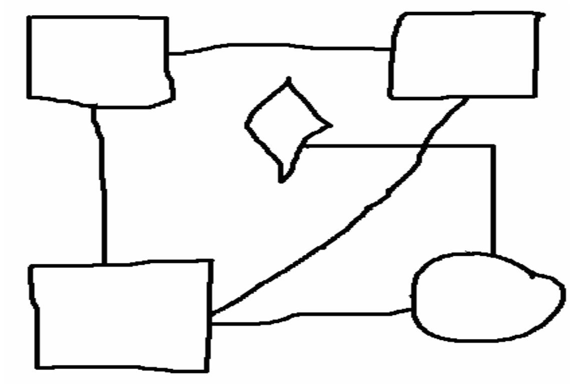


Fig. 7. Input Sample No 2

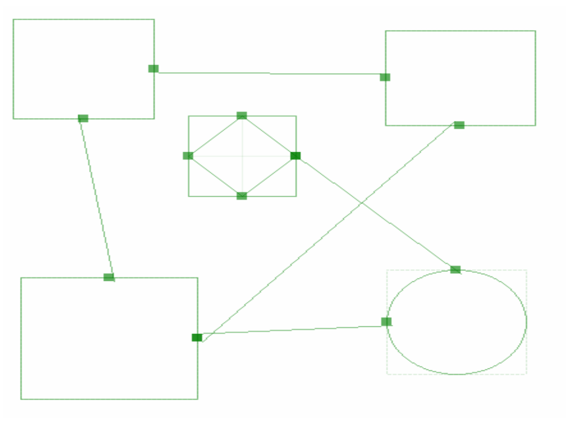


Fig. 8. Output Sample No 2

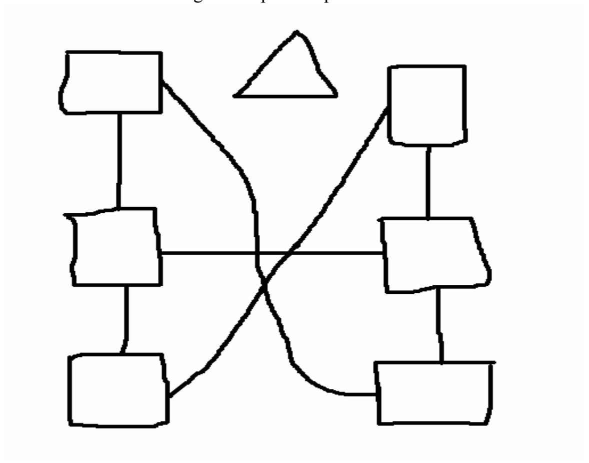


Fig. 9. Input Sample No 3

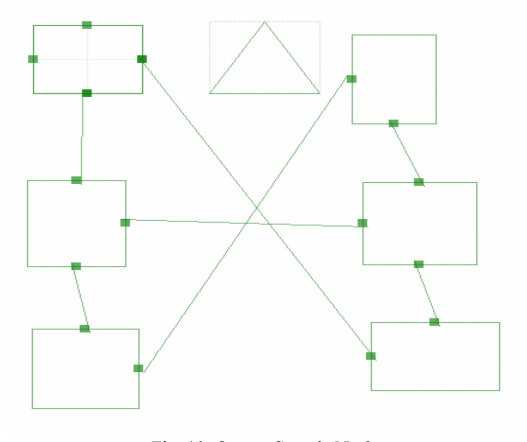


Fig. 10. Output Sample No 3

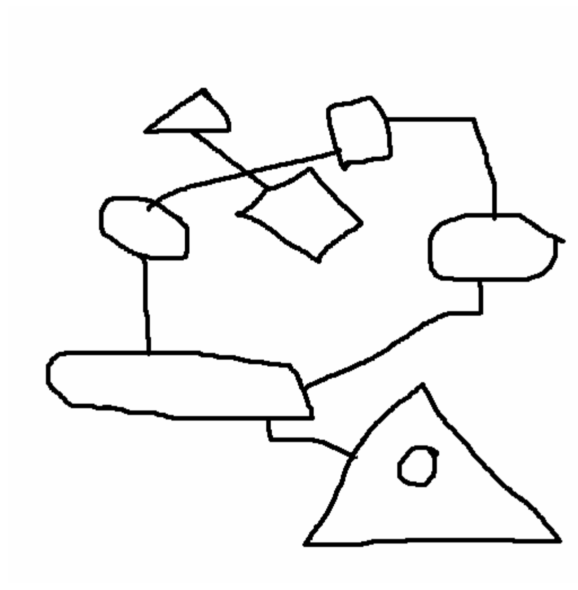


Fig. 11. Input Sample No 4

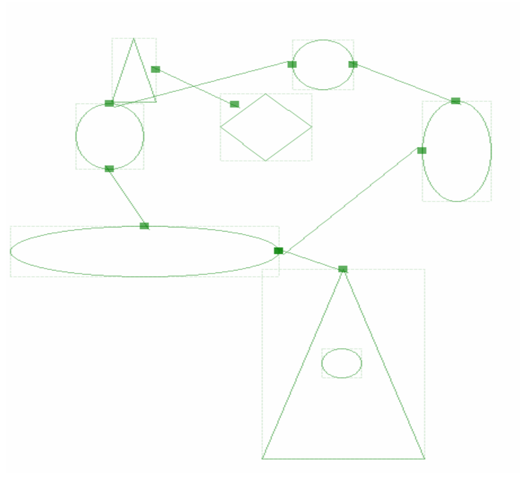


Fig. 12. Output Sample No 4

References

[1] Microsoft software product for creating a wide variety of business and technical drawings Available: www.office.microsoft.com.

[2] M. Notowidigdo, "User-Directed Sketch Interpretation (Thesis or Dissertation style)," MEng thesis, Massachusetts Institute of Technology, June 2004.

[3] B. Scholkopf and A. J. Smola, *Learning with Kernel*s. The MIT Press Cambridge, Massachusetts London, England 2002.

[4] R. Cao and C. L. Tan, "Line Primitive Extraction by Interpretation of Line Continuation," School of Computing, National University of Singapore.

[5] E. Lank and J. S. Thorley, "Interactive system for recognizing hand-drawn UML diagrams," *in Proc. of CASCON* 2000.

[6] T. Hammond and R. Davis, " A geometrical sketch recognition system for UML class diagrams," *In Proc. of AAAI Spring Symposium on Sketch Understanding,* 59.68.

[7] J. Lin, M. W. Newman, J. I. Hong, and J. A. Landay, "finding a tighter fit between tools and practice for web site design," *in Proc. of CHI '00, 510. 517.*

[8] D. Rubine, "Specifying gestures by example," in *Proc. of SIGGRAPH '91*, 329.337.

[9] T. M. Sezgin and R. Davis, "Early processing in sketch understanding," *in proc. of 2001 Perceptive User Interfaces Workshop*.

[10] J. A. Landay and B. A. Myers, "Interactive sketching for the early stages of user interface design," *in proc. of CHI* '95, 43.50.

[11] J. A. Jorge and M. Fonseca, "A simple approach to recognize geometric shapes," *GREC, LNCS 1941*, 1999.

[12] H. Hse and A. R. Newton, "Sketched Symbol Recognition using Zernike Moments," *Proc. of the 17th International Conference* Aug. 2004.

[13] E. Valveny and E. Marti, "Deformable template matching within a Bayesian framework for hand-written graphic symbol recognition," *GREC, LNCS 1941,* 1999.

[14] J. Y. Ramel, "A structural representation adapted to handwritten symbol recognition" *GREC, LNCS 1941,* 1999.

[15] M. Notowidigo and C. Miller "Offline sketch interpretation," American association of Artificial intelligence, 2004.

[16] R. C. Gonzalez, and R. E Woods, *Digital Image Processing 3rd Edition.*, hardcover Aug. 21, 2007.

[17] M. Szummer and Y. Qi, "Contextual Recognition of Hand-drawn Diagrams with Conditional Random Fields," Microsoft Research Oct. 2004.

[18] M. J. Fonseca and J. A. Jorge, "Using Fuzzy Logic to Recognize Geometric Shapes Interactively," *Proc. of the 9th Int. Conference on Fuzzy Systems* (FUZZ-IEEE 2000), Departamento de Engenharia Inform´atica, IST/UTL Av. Rovisco Pais, 1049-001 Lisboa, Portugal.

[19] J. E. Boyce, D. P. Dobkin, R. L. (Scot) Drysdale, and L. J. Guibas "Finding External Polygons," *ACM O-89791-067-2/82/O05/0282* 1982.

[20] Y. Chen and G. Zhang, "An improved method to the SVM multi-class classifier based on pairwise coupling," Machine *Learning and Cybernetics*, *2003* International Conference on Volume 5, Issue, 2-5 Nov. 2003.

[21] J. Arenas-Garcia and F. Perez-Cruz, "Multi-class support vector machines: a new approach," *Proc. ICASSP apos; 03* IEEE International Conference on Volume 2, April 2003.

[22] S. Xia, J. Li, L Xia and C. Ju, "Tree-Structured Support Vector Machines for Multi-class Classification," *ISNN*, vol 4493, 2007.