Robust Sketched Symbol Fragmentation using Templates

Heloise Hse
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720, U.S.A.
+1 510 642 2481
hwawen@eecs.berkeley.edu

Michael Shilman
Microsoft Research
One Microsoft Way
Redmond, WA 98052
+1 425 722 5853
shilman@microsoft.com

A. Richard Newton
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720, U.S.A.
+1 510 642 5771
newton@eecs.berkeley.edu

ABSTRACT
Analysis of sketched digital ink is often aided by the division of stroke points into perceptually-salient fragments based on geometric features. Fragmentation has many applications in intelligent interfaces for digital ink capture and manipulation, as well as higher-level symbolic and structural analyses. It is our intuitive belief that the most robust fragmentations closely match a user's natural perception of the ink, thus leading to more effective recognition and useful user feedback. We present two optimal fragmentation algorithms that fragment common geometries into a basis set of line segments and elliptical arcs. The first algorithm uses an explicit template in which the order and types of bases are specified. The other only requires the number of fragments of each basis type. For the set of symbols under test, both algorithms achieved 100% fragmentation accuracy rate for symbols with line bases, >99% accuracy for symbols with elliptical bases, and >90% accuracy for symbols with mixed line and elliptical bases.

Categories and Subject Descriptors
I.4.6 [**Image Processing and Computer Vision**]: Segmentation – *edge and feature detection.*

General Terms
Algorithms, Human Factors.

Keywords
Curve segmentation, perceptual grouping, shape templates, fitting, sketch-based user interface, HCI

1. INTRODUCTION
Sketching is a simple and natural mode of expression. It is especially desirable for conceptual design, both on an individual basis and in a collaborative environment. With a sketch-based user interface, one can have the freedom of sketching on paper and the benefit of an electronic design tool [9]. If a sketch system also includes a recognition capability, sketches can be interpreted and augmented with semantics so that they can be edited easily, efficiently searched, and neatened.

There has been a significant amount of research to date in various aspects of sketch-based user interfaces: interactive design tools [12, 13], studies of gestures [15], software toolkits [10], ink beautification [11], and sketch recognition [1, 24]. However, relatively little work has focused on the fragmentation of hand-sketched symbols (e.g. [4, 21, 25]). Fragmentation is a perceptual analysis of ink strokes in which stroke points are clustered into geometrically salient primitives, such as line segments and elliptical arcs. Figure 1 shows an example fragmentation of a sketched square and the ways that the resulting fragmentation can be utilized.

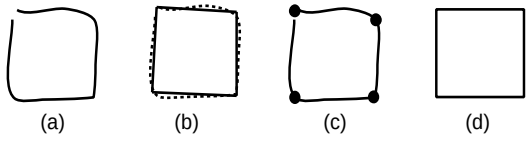

Figure 1. (a) An initial stroke, (b) its natural fragmentation (overlaying the stroke points), (c) a direct manipulation user interface based on the fragmentation, (d) a beautification of the stroke.

Fragmentation is a very basic problem, making it widely applicable to intelligent ink manipulation as well as other higher-level digital ink analyses. The structural information that it generates can be useful in the following situations:

- Generating structural descriptions for use in symbol recognition, especially by structural recognizers [4, 17].
- Locating functional points in a symbol, such as the tip of an arrowhead, the four corners in a square, etc. (Figure 1c) This is especially applicable to recognizers that do not perform structural analysis [2, 8, 14, 20, 25], such as statistical recognizers [8, 20, 25].
- Automatically 'neatening' a symbol using the geometric primitives which result from fragmentation (Figure 1b). Further beautification of the symbol can be based on the recognition result or geometric properties such as parallel lines, right angles, etc. (Figure 1d) [11]

IUI'04, January 13–16, 2004, Madeira, Funchal, Portugal.

- Providing a direct manipulation user interface with which to interact with sketched ink (Figure 1c). For example, Saund's PerSketch provides fluid interaction with sketch fragments based on perceptual salience [22].

Most segmentation techniques in image processing target nicely printed symbols and do not scale well to hand-sketched symbols due to the noise and distortion in sketches. Sezgin [23], and later Calhoun [4], used both curvature and speed information in a stroke to locate breakpoints. Yu applied the mean shift procedure to approximate strokes [27]. Saund's approach to fragmentation uses more perceptual context, including local features such as curvature and intersections, as well as more global features such as closed paths [21]. However each of these methods is still susceptible to over- and under-fragmentation of strokes. Their methods use empirical thresholds to test the validity of an approximation which ultimately leads to the problem of a threshold being too tight or too loose. A noisy stroke that just falls outside the threshold will not be fragmented correctly.

In this paper, we describe a robust fragmentation approach that generates consistent and correct structural descriptions of symbols such that the resulting fragmentations closely match human perception. Fischler and Bolles [7] suggest that humans have a rich visual vocabulary of iconic primitives with which to impart structure to data. When given a geometric symbol, the human visual system easily identifies an optimal fragmentation, even in the presence of noise, based on a mental model of that symbol.

We build on this observation to create a simple and efficient fragmentation model based on templates. The first type of fragmentation template is an ordered sequence consisting of line segments and elliptical arcs. Given a sketch and a template, our algorithm determines the optimal set of breakpoints for the fragmentation of the sketch according to the template. The second type of template consists of only the number of line segments and the number of elliptical arcs and is equivalent to the first template without the ordering information. Both methods are simple and are free of empirical thresholds.

In Section 2, we elaborate on our choice of line segments and elliptical arcs as basis geometries. In sections 3 and 4, we present the two template-based fragmentation algorithms based on a *Dynamic Programming* (DP) approach. Implementation and results are discussed in Section 5.

2. BASIS FRAGMENTS

The choice of basis representation for fragmentation is dependent on the application. Our objective is to fragment a wide variety of sketched symbols (e.g. squares, ovals, trapezoids, pentagons, etc.) into simpler structures such that they are both faithfully represented and less complex than their original form. There are a number of published approaches to polygonal approximation in which planar curves are fragmented into line segments. While such approaches make approximation simpler, since only a first-degree polynomial is used, it is ineffective to represent smooth curves with lines. Curves can be approximated using higher order primitives such as circular arcs [26], ellipses [16] and splines [5], but significantly more computation is required when these additional parameters are involved. In this work, we chose to use segmented ellipses in our basis over circular arcs because we have found such approximation provides a more concise and natural fitting result for the family of symbols we are working with now and are likely to work with in the future. Although higher order polynomials (e.g. splines) can be used to interpolate and/or approximate a set of points, they are far less likely to provide useful structural information for later use in symbol recognition.

It should also be noted that the algorithms introduced in this paper are not limited to fragmentation with line and ellipse bases. Rather, these algorithms provide a general structure for optimal fragmentation, and any basis can be easily substituted.

3. OPTIMAL FRAGMENTATION USING ORDERED TEMPLATES

There have been several earlier attempts to produce optimal solutions for curve partitioning with constraints [3, 18]. Bellman was the first to use DP for curve approximation with line segments [3] given that the analytic expression of the curve is known. Perez [18] used a DP approach to optimally approximate a digitized curve with a given number of line segments. Our method extends the basis of primitives from lines to lines and elliptical arcs. In addition, our algorithms can be applied to multiple strokes, not just one. For the remainder of this paper, we will use the word basis to mean either a line segment (L) or an elliptical arc (E).

If the stroke order of a sketched symbol is known, the template can be described as an ordered sequence of L's and E's. For example, the template for the character symbol 'D', in which the vertical line is drawn before the elliptical arc, is 'LE'. Of course, if a symbol consists of only one type of basis, ordering is not an issue. In this section, we present an algorithm that optimally fragments a given sequence of strokes with this form of template (an ordered sequence of L's and E's).

3.1 Problem Formulation

Given a sketched symbol S and a template T, find a set of breakpoints in S such that the fitting performed according to T yields the minimum fit error. The sketched symbol S consists of a sequence of strokes $\{S_1, S_2, ..., S_N\}$ and each stroke S_i contains a sequence of timed-ordered points $\{P_i^1, P_i^2, ..., P_i^M\}$. The template T is a string of L's and E's. For example, the template for squares would be LLLL and the template for P's would be LE or EL depending on the stroke order. The number of breakpoints needed to be identified is $K=T.\text{len}-N$, where T.len is the number of basis fragments in T, and N is the total number of strokes. The fragmentation algorithm requires symbols to contain fewer strokes than the number of basis elements to fit.

3.2 A Dynamic Programming Algorithm

The problem of "fitting to a template" is an optimization problem in which the goal is to minimize the error from fitting a shape with basis elements by identifying an optimal set of breakpoints. Suppose k breakpoints are needed to fragment S into T, a brute-force approach would do an exhaustive search on all combinations of k breakpoints. This approach requires testing mCk (m choose k) sets of breakpoints, where m is the total number of data points in S. The number of combinations is exponential in the size of m, and therefore this exhaustive search method is a poor strategy and not practical for use in interactive applications.

$$d(n,m,k,t)=\begin{cases}\left[\sum_{i=1}^{n-1} f(S_i,0,M_i,t[i])\right]+f(S_n,0,m,t[n]) & ;\ \text{if } k=0\\ \min_{k<i<m}\{d(n,i,k,t[1..t.\text{len}-1])+f(S_n,i,m,t[t.\text{len}])\} & ;\ \text{if } n=1,k>0\\ \min\begin{Bmatrix} d(n-1,M_{n-1},k,t[1..t.\text{len}-1])+f(S_n,0,m,t[t.\text{len}]),\\ \min_{1<i<m}\{d(n,i,k-1,t[1..t.\text{len}-1])+f(S_n,i,m,t[t.\text{len}])\}\end{Bmatrix} & ;\ \text{if } n>1,k>0\end{cases}$$

Figure 2. Optimal fragmentation of a sequence of strokes, $\{S_1, \ldots, S_n\}$ to a sequence of E's and L's using DP.

$$d(n,m,k,e,l)=\begin{cases} d(n,m,k,t),\ t \text{ is a string of } e \text{ E's} & ;\text{if } l=0\\ d(n,m,k,t),\ t \text{ is a string of } l \text{ L's} & ;\text{if } e=0\\ \min\begin{Bmatrix}\min_{k<i<m}\{d(n,i,k-1,e,l-1)+f(S_n,i,m,\text{'L'})\},\\ \min_{k<i<m}\{d(n,i,k-1,e-1,l)+f(S_n,i,m,\text{'E'})\},\end{Bmatrix} & ;\text{if } n=1,\ l>0,\ e>0\\ \min\begin{Bmatrix} d(n-1,M_{n-1},k,e,l-1)+f(S_n,0,m,\text{'L'}),\\ d(n-1,M_{n-1},k,e-1,l)+f(S_n,0,m,\text{'E'}),\\ \min_{1<i<m}\{d(n,i,k-1,e,l-1)+f(S_n,i,m,\text{'L'})\},\\ \min_{1<i<m}\{d(n,i,k-1,e-1,l)+f(S_n,i,m,\text{'E'})\}\end{Bmatrix} & ;\text{if } n>1,\ l>0,\ e>0\end{cases}$$

Figure 3. Optimal fragmentation of a sequence of strokes, $\{S_1, \ldots, S_n\}$ to a set of E's and L's using DP.

Below, we describe a polynomial time algorithm that is simple and optimal using a DP approach.

First, we define the optimal substructure for the fragmentation problem. An optimal fragmentation of S that chooses a breakpoint at P_i^j contains the optimal fragmentation of the stroke(s) up to P_i^j. In other words, to find an optimal fragmentation of S with template T, one assumes that the optimal solution for fragmenting everything up to P_i^j with a template $T[1...T.\text{len}-1]$ has been computed, and the piece from P_i^j to the end is then fit with $T[T.\text{len}]$.

Next, a recursive solution is defined based on this optimal substructure. Let $d(n,m,k,t)$ be the minimum fitting error to approximate every point up to the m^{th} point in the n^{th} stroke with the template t, and let $f(S_n,i,m,t[j])$ be the fitting error resulting from fitting the segment from P_n^i to P_n^m using $t[j]$. If $t[j]$ is 'E', elliptical fitting on the data points is performed [19]; if $t[j]$ is 'L', total least square fitting is performed [6]. The best fragmentation for S with N strokes using K breakpoints and a template T would thus be $d(N,M_N,K,T)$ where M_N is the index of the last point in S_N.

$d(n,m,k,t)$ is defined as follows. When k=0, each of the first (n–1) strokes is fit with the corresponding basis in the template and the segment from P_n^0 to P_n^m in the n^{th} stroke is fit with the last primitive in the template. When n=1 and k>0, a choice has to be made on a point P_n^i to be the breakpoint and i>k, otherwise the number of breakpoints required would exceed the number of data points available. When n>1 and k>0, in addition to checking the best breakpoint to use in S_n, the previous stroke (S_{n-1}) must also be checked because it is possible that the best breakpoint may lie in any of the previous strokes. Due to the optimal substructure, the optimal fragmentation for the last point in the previous stroke S_{n-1} is all that must be checked. The recursive definition for fragmentation of S is given in Figure 2. The algorithm has a run time complexity of $O(K\times M^2)$ where K is the number of breakpoints and M is the total number of data points. The space requirement is $O(K\times M)$ for keeping a table of solutions to the sub-problems.

4. OPTIMAL FRAGMENTATION USING UNORDERED TEMPLATES

If the basis ordering information is not known (i.e. only the number of each type of basis element, "E" or "L", are passed into the fragmentation routine), the fragmentation problem becomes more complex. In a naïve approach, the algorithm presented in the previous section could be applied to each combination of the basis elements. The combination that yields the least fit error is selected as the optimal fragmentation of the sketched symbol. For a template consisting of l L's and e E's, there are a total of $(l+e)Ce$ number of orderings. This number is exponential in the size of l and e. Using DP, this problem can be solved in polynomial time $O(K\times M^2\times l\times e)$. This result is more efficient than using DP to solve all $(l+e)Ce$ combinations since the solutions to each of the overlapping sub-problems is only computed once. Given a set of strokes and the number of lines (l) and ellipses (e), Figure 3 shows a recursive solution based on DP for optimal fragmentation of the set of strokes.

5. IMPLEMENTATION AND DISCUSSION

Our target class of application for this work is one that has a bounded set of target symbols from which to select (e.g. a UML diagram editor, a slide drawing program like Microsoft PowerPoint, or an electrical schematic editing tool). We have implemented both algorithms and tested them on user sketched symbols collected from 17 users. Each user was asked to sketch ≈30 examples for each of the 10 symbols shown in the top two

rows of Figure 4. The data set contains a total of 5,928 examples overall and about 540 examples per symbol. The first algorithm using ordered template was evaluated against human perception of breakpoints. A correct fragmentation consists of breakpoints in the places where one would expect them to be. The second algorithm using unordered templates is evaluated by checking both the breakpoint placement and the correctness of the templates it generates. For all symbols, except crescents and arches, both algorithms achieved 100% accuracy rate in identifying the correct set of breakpoints. For crescents and arches, the algorithms were able to achieve over 99% fragmentation accuracy. To further evaluate the algorithms on symbols consisting of mixed line segments and elliptical arcs, three more symbols are introduced (cylinders, callouts and plaques shown in the bottom row of Figure 4). We collected these data from 8 users and obtained ≈150 examples per symbol. The overall results are shown in Table 1.

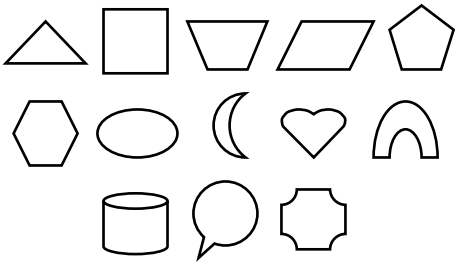

Figure 4. Symbol set used in evaluation: triangle, square, trapezoid, parallelogram, pentagon, hexagon, oval, crescent, heart, arch, cylinder, callout, and plaque.

Table 1. Fragmentation accuracy result for both algorithms

	Ordered templates (%)	Unordered templates (%)
Triangles	100	100
Squares	100	100
Trapezoids	100	100
Parallelograms	100	100
Pentagons	100	100
Hexagons	100	100
Ovals	100	100
Crescents	99.62	99.62
Hearts	100	100
Arches	99.82	99.08
Cylinders	98.04	95.42
Callouts	99.34	99.34
Plaques	93.38	90.07

From the experimental data, it is evident that both fragmentation algorithms are robust. The analysis on mis-fragmented examples showed that the breakpoints were mis-identified at connection points of a line segment and an elliptical arc due to the fact that the transition from one to the other is often blended and indistinguishable, especially under quick pen motion. Figure 5 shows the result of the ordered-template algorithm performed on seven sketched symbols. The circles on a symbol indicate the breakpoints computed by the algorithm. We consider breakpoints to be interior points in a stroke, and therefore the first and last points of a stroke are not considered to be breakpoints, even though they may form a corner. All of our experiments were performed on symbols in their original sketched form without any preprocessing (e.g. de-hooking, point reduction, smoothing, etc.)

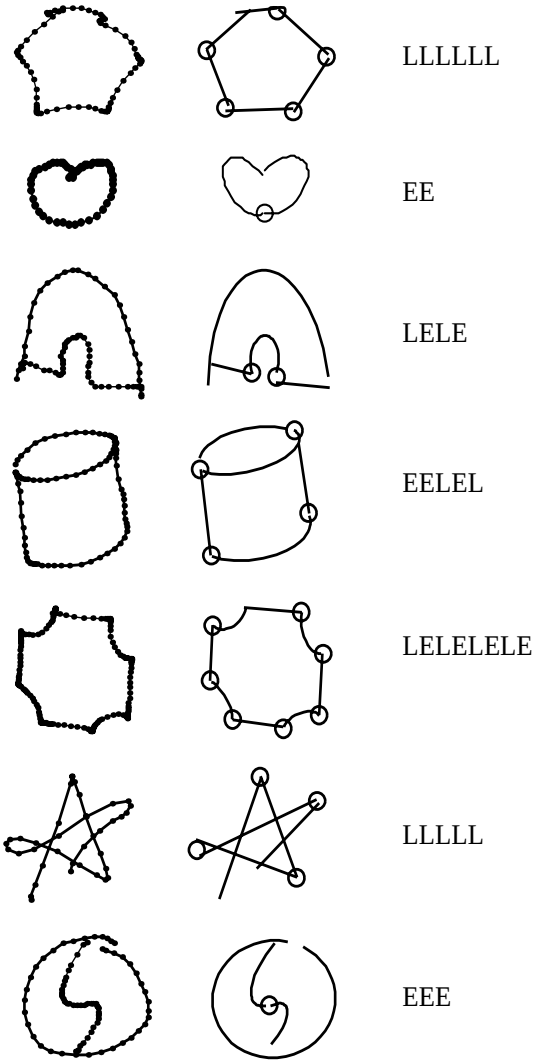

Figure 5. Sketched symbols, fragmentations, and templates.

When no information about the number of L's or E's is given, the fragmentation problem becomes less tractable. Given an error threshold, an optimal fragmentation that uses the fewest number of segments to achieve an approximation error below the threshold can be obtained by testing with 1 L, 1 E, 2 L's, 2 E's, 1 L and 1 E, and so forth, using the aforementioned algorithm with a different unordered template each time. The complexity of this approach is exponential, which is not practical for real-time

applications. This problem is similar to the general curve partitioning problem since no information about L's and E's is given. Many heuristic methods have been developed to address the problem in image processing, however these methods are highly problem dependent (e.g. industrial parts inspection, map contour approximation in images) and do not adapt well to fragmenting sketched symbols due to distortion and imprecise nature of sketches. For example, in hand-sketched symbols, corners may not always be sharp, lines may not be straight, and curves may not be smooth. If some information such as ordered bases, unordered bases, or even the number of fragments can be obtained, the fragmentation problem becomes much more tractable and our DP algorithms provide a robust method for fragmentation.

6. CONCLUSION

In this paper, we considered two types of fragmentation templates. One specifies a sequence of lines and ellipses and the other specifies the number of lines and ellipses. The methods, based on dynamic programming, are efficient and robust. The result shows that the fragmentations closely match human perception. The fragmentation accuracy rate is 100% for the symbols with line bases, >99% for the symbols with elliptical bases, and >90% for the symbols with mixed line and elliptical bases.

7. REFERENCES

[1] Alvarado, C., Oltmans, M. and Davis, R., A Framework for Multi-domain Sketch Recognition. in 2002 AAAI Spring Symposium - Sketch Understanding, (Palo Alto CA, 2002), AAAI Press, 1-8.

[2] Apte, A., Vo, V. and Kimura, T.D., Recognizing Multistroke Geometric Shapes: An Experimental Evaluation. in UIST 1993, (Atlanta Georgia, 1993), ACM Press, 121-128.

[3] Bellman, R. On the Approximation of Curves by Line Segments using Dynamic Programming. Commun. ACM, 4 (6). 284.

[4] Calhoun, C., Stahovich, T.F., Kurtoglu, T. and Kara, L.B., Recognizing Multi-Stroke Symbols. in 2002 AAAI Spring Symposium - Sketch Understanding, (Palo Alto CA, 2002), AAAI Press, 15-23.

[5] Chang, H. and Yan, H. Vectorization of Hand-drawn Image using Piecewise Cubic Bezier Curves Fitting. Pattern Recognition, 31 (11). 1747-1755.

[6] Duda, R.O. and Hart, P.E. Pattern Classification and Scene Analysis. Wiley Press, New York, 1973.

[7] Fischler, M.A. and Bolles, R.C. Perceptual Organization and Curve Partitioning. IEEE Trans. PAMI, 8 (1). 100-105.

[8] Fonseca, M.J., Pimentel, C. and Jorge, J.A., CALI: An Online Scribble Recognizer for Calligraphic Interfaces. in 2002 AAAI Spring Symposium - Sketch Understanding, (Palo Alto CA, 2002), AAAI Press, 51-58.

[9] Hearst, M.A., Gross, M.D., Landay, J.A. and Stahovich, T.F. Sketching Intelligent Systems. IEEE Intelligent System, 3 (3). 10-19.

[10] Hong, J.I. and Landay, J.A. SATIN: A Toolkit for Informal Ink-based Applications. CHI Letters: ACM Symposium on User Interface Software and Technology: UIST 2000, 2 (2). 63-72.

[11] Igarashi, T., Matsuoka, S., Kawachiya, S. and Tanaka, H., Interactive Beautification: a Technique for Rapid Geometric Design. in UIST 1997, (Canada, 1997), 105-114.

[12] Landay, J.A. and Myers, B.A. Sketching Interfaces: Toward More Human Interface Design. IEEE Computer, 34 (3). 56-64.

[13] Lin, J., Newman, M.W., Hong, J.I. and Landay, J.A. DENIM: Finding a Tighter Fit between Tools and Practice for Web Site Design. CHI Letters: ACM Symposium on User Interface Software and Technology: UIST 2000, 2 (1). 510-517.

[14] Lipscomb, J.S. A Trainable Gesture Recognizer. Pattern Recognition, 24 (9). 895-907.

[15] Long, A.C., Landay, J.A. and Rowe, L.A. Visual Similarity of Pen Gestures. CHI Letters: ACM Symposium on User Interface Software and Technology: UIST 2000, 2 (1). 360-367.

[16] Pavlidis, T. Curve Fitting with Conic Splines. ACM Trans. on Graphics, 2 (1). 1-31.

[17] Pavlidis, T. Structural Pattern Recognition. Springer-Verlag Press, Berlin, 1977.

[18] Perez, J. and Vidal, E., An Algorithm for the Optimum Piecewise Linear Approximation of Digitized Curves. in 11th IAPR International Conference on Pattern Recognition, (1992), 167-170.

[19] Pilu, M., Fitzgibbon, A. and Fisher, R. Direct Least-Square Fitting of Ellipses. IEEE Trans. PAMI, 21 (5). 476-480.

[20] Rubine, D. Specifying Gestures by Example. SIGGRAPH '91, 25 (4). 329-337.

[21] Saund, E. Finding Perceptually Closed Paths in Sketches and Drawings. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (4). 475-491.

[22] Saund, E. and Moran, T., A Perceptually Supported Sketch Editor. in UIST 1994, (Marina del Rey CA, 1994), ACM Press, 175-184.

[23] Sezgin, T.M., Stahovich, T. and Davis, R., Sketch Based Interfaces: Early Processing for Sketch Understanding. in PUI 2001, (Orlando FL, 2001), ACM Press.

[24] Shilman, M., Pasula, H., Russell, S. and Newton, A.R., Statistical Visual Language Models for Ink Parsing. in 2002 AAAI Spring Symposium - Sketch Understanding, (Palo Alto CA, 2002), AAAI Press, 126-132.

[25] Ulgen, F., Flavell, A. and Akamatsu, N., Recognition of On-Line Handdrawn Geometric Shapes by Fuzzy Filtering and Neural Network Classification. in HCI International '95, (Yokohama, Japan, 1995), 567-572.

[26] West, G. and Rosin, P. Techniques for Segmenting Image Curves into Meaningful Descriptions. Pattern Recognition, 24 (7). 643-152.

[27] Yu, B., Recognition of Freehand Sketches using Mean Shift. in IUI 2003, (Miami FL, 2003), 204-210.