Sketch Recognition using Particle Swarm Optimizations

Maha El Meseerya, Mahmoud Fakhr El Dina, Samia Mashalia, Magda Fayekb, Nevin Darwishb

aSignals Processing Group, Computers and Systems Department, Electronic Research Institute, Cairo, Egypt

melmeseery@eri.sci.eg, mafakhr@mcit.gov.eg, [samia@eri.sci.eg](mailto:samia@eri.sci.eg)

bComputer Engineering Department, Faculty of Engineering, Cairo University, Cairo, Egypt

magdafayek@gmail.com, [ndarwish@ieee.org](mailto:ndarwish@ieee.org)

**Abstract**

Sketches and drawings are widely used as simple methods of expressing thought and designs. The goal of this paper is developing a sketch recognition system that facilitates using sketches in computer systems. The system uses Particle Swarm Optimization (PSO) algorithm to optimally segment the strokes the user draws into meaningful geometric primitives. These geometric primitives are grouped to form symbols which are further identified using a Support Vector Machines (SVM) classifier. Two versions of PSO segmentation algorithms were implemented. Experiments were conducted on three different dataset, simple presentation (Hs-DS), Electrical Design (EL-DS) and Logic Design (LD-DS) symbols. Results show that both PSO algorithms produce better segmentation than most current systems. This segmentation improvement contributed in getting more accurate recognition results on all the datasets tested. The results prove that the system is effective specially in datasets with low diversity in shapes, i.e., shapes are very similar, as in Logic Design dataset (LD-DS).

Symbol Recognition, Particle Swarm Optimization, Sketch Recognition, Geometrical Features, Polygon approximation

**1** **Introduction**

Computer research nowadays aims at facilitating computer human interaction in every way. Pen-based interfaces give the user a pencil-paper like feeling that enhances interaction more than the current used keyboard-mouse computer interfaces. However, using sketches as input to a computer system requires the system to understand the drawn sketches. In a truly freely sketch system this is not a trivial problem due to the sloppy and messy way in which users draw sketches. Sketches here stand for any hand drawing the user draws using digital stylus or similar device. Normally, sketches are composed of a set of strokes. The time between pen-up and pen-down events identifies each stroke. The strokes are mainly the path of points, which the user draws between the pen-down and pen-up events while using the stylus or similar device.

Sketch recognition can be applied either using Pattern Recognition (PR) based method or Artificial Intelligence (AI) based method. In AI based methods, Symbols are represented using hierarchical levels of simpler geometric shapes or primitives and spatial relations between them. Systems that employ such representation must first pre-process strokes to break them into geometric primitives. Recognition can then is treated as an AI problem which can be solved using methods like sub graph matching, Bayes belief network, and constrain satisfaction. A Hierarchical shape descriptions was widely used in sketch systems [14, 2].

Recognition based on AI methods as applied in [2] has the principle problem of computational complexity of the matching between shape description and geometric primitives produced in the segmentation process. Natural sketch are difficult to be segmented and grouped to create reliable fragment of geometric primitives. This is partially due to the increase of noise and other phenomena such as overlapped, over traced strokes and sharing of strokes between different symbols. Since these systems are dependable on the segmentation results, they either perform recognition using undependable segmentation results or consider different ways to segment the input sketch. This is because they assume that each stroke plays a role in the structure of the shape. In PR based methods, systems compute properties that describe symbols then use a classification method to recognize symbols. The properties used can either be based on gesture properties, global shape properties or image based properties. Gesture properties are set of features that represent how the user draws a symbol not how it looks like (e.g., identify a rectangle when a user draws a L corner). In gesture properties approach, systems identify symbols by using gesture properties of some of its underlying strokes. System that uses gestures properties assume that users draw symbols in specific order and style [20, 18]. In global shape properties approach, systems investigate general shape properties and its underlying strokes. Using shape properties (e.g., ratio of the area of convex hull to the area of shape) relaxes the assumption and the role of each stroke in the symbol but it does not truly represent the appearance or fine details of the symbol [19, 8]. Unlike shape properties based methods, image based methods disregard individual strokes and focus on the overall appearance of shapes. These system convert the shapes into image and extracted image based features (e.g. local gradient feature)[17, 13]. In [13], they used four different distance measures to match user symbol with the prototype database. The prototype of a shape is represented as an image of the symbol that is normalized and scaled. However, the problem is that the shape prototype must represent all transformation and variation the input shape can have. Methods based on gesture and global shape properties produce systems that are dependent on assumptions about how strokes can be grouped. These systems assume that users will finish drawing a shape before moving to the next one [8, 10]. This assumption allows the system to temporally group strokes and then recognizes them based on their strokes or global shape properties. Alvarado [3] argued that this assumption does not hold in natural sketches.

In this paper, we choose to implement a hybrid method between shape property, gesture and hierarchical (AI method) based approaches. Our method attempts to avoid many of the problems faced by approaches based on individual gestures or global shape properties while trying to gain the additional information that the stroke based system uses. Firstly, the system uses particle swarm optimization (PSO) to optimally segment user's strokes. Users can draw symbols using any number of strokes in any order they like. The system segments each stroke using two different PSO algorithms. The first segmentation algorithm is based on dividing the digital curve into polygons[22]. Our algorithm enhances its performance by adding curvature and other local information while segmenting strokes to achieve better results. To handle complex strokes, our system uses a second segmentation PSO algorithm to segment strokes into a set of lines and circular arcs. After segmentation, a feature vector composed of different statistical, geometrical and composite features is deduced. The final classification step uses a SVM classifier to classify symbols into one of the previously trained classes based on the features vector computed.

The remaining of the paper is as follows; the literature review on stroke segmentation is presented in Section 2. Section 3 introduces Particle Swarm Optimization. Section 4 describes the proposed system in details. Section 5 describes the experiments that were preformed. Finally, the conclusion and future work is presented in Section 6.

**2** **Literature Review**

Yu [23] introduced a *feature area* for each primitive and then computed the segmentation error for different types of primitives based on the computed *feature area*. His system achieved good accuracy in simple shapes (square, ellipse,...etc) but did not perform well in complex shapes. A hybrid algorithm was introduced in [21][[1]](#footnote-2) where different sets of segments are generated based on both curvature and speed dominant points, followed by choosing a segmentation with the least error from a generated hybrid set. However, this system is limited to recognizing only specific simple geometric shapes with a set of low level recognizers. Each low level recognizer is designed to recognize only one geometric shape using spatial and geometric information extracted from input stroke.

A genetic algorithm was used by [6] to optimally divide digital curves into lines and curves. Chen et al. [6] uses digital curves scanned from paper as input to the system. However, they did not take the advantage of the curvature or local geometric properties of the digital curve. Yin [22] used PSO to convert digital curves into polygons. Our system adopts Yin [22] method and tries to improve it by adding curvature and other local information while segmenting strokes to achieve better segmentations. The next section presents the general particle swarm algorithm which is used in this system.

**3** **Discrete Particle Swarm Optimization**

The main idea of *Particle Swarm Algorithm (PSO)* is to represent each solution with a dimension particle from the solution space [15]. Each particle moves with a direction and velocity based on equations 1 & 2.

(1)

where represent the th dimension in the th particle and is the velocity of the th dimension in the th particle.

(2)

where is the inertia weight parameter which controls the tradeoff between exploration and exploitation, is the local best particle, is the global best particle, & are random variables and & are the swarm acceleration parameters.

After each iteration the global best particle and the agent local best particle are evaluated based on the maximum fitness functions of all particles in the solution space. The solution is found after achieving a specific number of iteration or after an error threshold is achieved. Equation 3 is used to change the general swarm algorithm into binary particle (*Discrete Particle Swarm Algorithm DPSO*). The *DPSO* handles particle values of either or [16].

(3)

where is the numerical values of the particle and is a random variable.

**4** **The Proposed System**

The suggested sketch recognition system is divided into three main steps; 1) preprocessing, 2) segmentation and 3) Recognition. The following section provides a detailed description of each step.

**4.1** **Preprocessing**

Speed and time difference information were widely used in sketch recognition systems [21]. Figure 1 shows an example of an input stroke drawn by users. Figure 2 shows the graph of time difference and the speed for the stroke in Figure 1. It was noted that time difference provides more distinct maxima than the speed information. Agar et al. [1] mentions that the pointing device (e.g., mouse) sampling rate is the reason for this phenomenon. The points are sampled at regular time intervals as long as the pointing device is moving. Therefore, there are no new detected points when the pointing device is stationary. This leads to a nearly constant time difference between samples while the pen is moving and large difference while the pen is stationary (Figure 1). Contrary to time difference information, the speed information, where the users draw with variant speeds, the speed information has a lot of noise in the data (Figure 2).

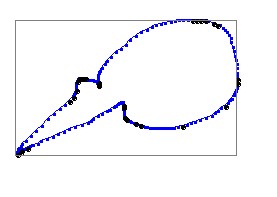


Figure 1: Input Stroke

Example of an input stroke to the system.

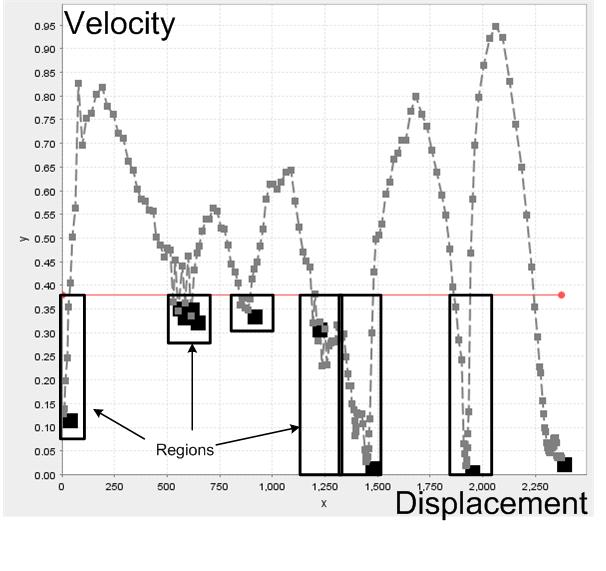
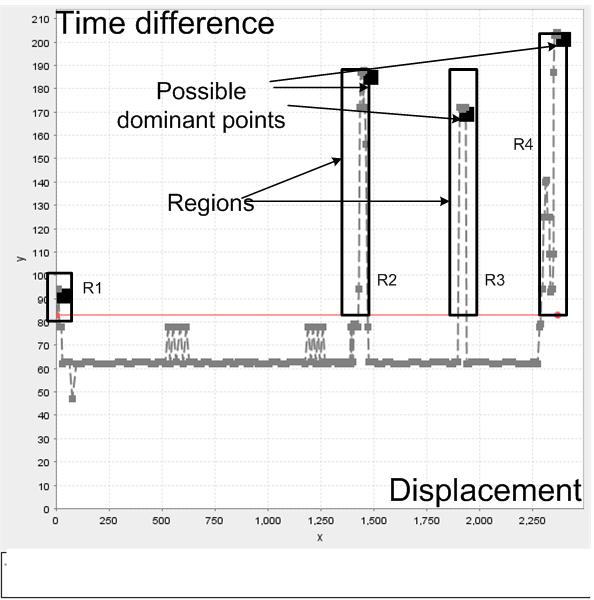
[Speed versus Displacement Graph]  [ Time Difference versus Displacement Graph] 

Figure 2: Speed and Time Difference Graphs

a) speed graph has lot of noise in the data with respect to b) time difference. The black points denote the location of the possible dominant points . Black rectangles represent regions that in a) Are lower than the threshold. b) Are above the threshold. Dominant points are characterized by low speed values, high curvature and direction values. However, the dominant points from the curvature information may not be the same points as the dominant points extracted from the speed information. Hence, we compute time difference, direction, speed and curvature of each point along a stroke to ensure that no dominant point is overlooked in this phase. The speed is calculated as where is the time difference between two points and is the length between them. The direction calculated as the angle between vectors and the and the curvature is considered as the change in direction with respect to length i.e. .

After the system computes the speed, time difference and curvature information, it proceeds to detect the points that have low velocity and high curvature. However, using simple differentiation to detect local extreme points results in false points due to the non smooth curves. Hence, our system adopted a process presented by [21], where the mean of each of the four calculated information is calculated. This mean is taken as the threshold *th* which is used to separate the curve into regions () (Figure1); each region is defined as a range of points, where the curve values are either above or below the threshold *th*. Those regions are further processed to find the maximum point of each region . The stroke points that correspond to those maximum values are labeled as *possible dominant points* . The system repeats this process for curvature, time difference and speed information. All the points labeled as possible dominant points are saved into a single array. Figure 3 shows the particles labeled as Possible dominate point in the preprocessing phase. It is clear that some of the points are redundant as specified in the preprocessing stage.

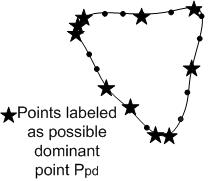


Figure 3: Possible Dominante Points

An illustration of stroke points and the expected *possible dominant points* .

**4.2** **Segmentation**

In the segmentation stage, the input stroke is divided into a set of primitives. As shown in Figure 4 first an attempt is made to fit the stroke points into a curve or an ellipse using a minimum square error fitting algorithm [9]. The ellipse fitting step mainly helps to prevent the system from over segmenting the stroke into multiple lines or curves if the input stroke is an ellipse. If the stroke proved to be an ellipse arc then the segmentation process ends and the system proceeds to the next step. Otherwise, the stroke is passed to two further segmentation algorithms that divide the stroke to either lines or lines and curves. Hence we generate two different segmentations; the system chooses the segmentation with the minimum error. The following sections describe in details the ellipse detection algorithm and the two segmentation algorithms used to divide the stroke.

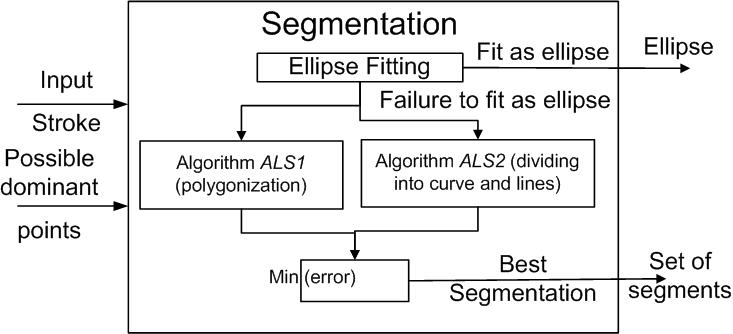


Figure 4: Segmentation Block

**4.3** **Ellipse fitting**

This process attempts to fit the stroke points into an ellipse arc; it starts with computing the center of the stroke bounding box. The bounding box center point is used as the first estimation of the center of the ellipse. The axes of the ellipse are estimated as the and of the stroke bounding box where and are the width and height of the bonding box. The least square fitting algorithm [7] is used to minimize the fitting error of the ellipse Equation (4)

(4)

where is number of points in the stroke, & are the length of ellipse axes, & are the coordinates of the center point and & are the coordinates of point in the stroke. A list of new values for , , and are generated randomly from the older values with small increments after each loop. After few iterations, the final fit error of the estimated ellipse is reported. Another measure is used to compute the efficiency of the final estimated ellipse. Equation 5 ensures that the drawn percentage of ellipse is considered. This eliminates fitting a line into very large ellipse but leaves small ellipses to be fitted as a partial or a full ellipse.

(5)

where is the error computed by Equation (4), and

where total length of stroke and is the perimeter of the estimated ellipse. If has a value more than threshold [[2]](#footnote-3) then the stroke is segmented as an ellipse otherwise the system proceeds to the next step.

**4.4** **Non ellipse fitting algorithm**

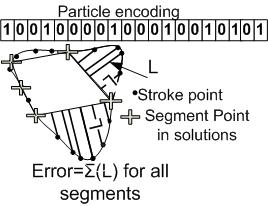


Figure 5: DPSO algorithm encoding

The swarm particle encoding where 1 represent the dominant points or segment points (i.e points are drawn as a big plus). Length L is the distance between a stroke point and the line connecting two successive segment points. Two DPSO algorithms are used to generate different segmentations for each stroke the user draws. For each segmentation generated, an error is evaluated. The segmentation with the minimum error value is chosen as the best stroke segmentation (Figure 4). The problem definition is the same in both algorithms but they differ in the method they use to compute fitness and error functions.

The input stroke with points can be represented by set where is the location of the point . The swarm algorithms consist of agents which are represented by the set where is a single solution particle from the solution space. Each particle decodes the problem using a binary array with the same length as the input stroke.

Therefore, the system represents each particle by where has only two values a)1 (); means that this point () is a dominate point, or b) 0 () means this point () is not a dominate point. Figure5 shows a particle encoding in the DPSO system.

The fitness function and error calculation are different in each of the two *DPSO* algorithms.

**4.4.1** ***Polygon Approximation Algorithm (AlgS1)***

The approximation error is computed by the equation 6. The graphical meaning of the error is shown in Figure5.

(6)

where the arc is defined as the consecutive set of points from point to point as in . The line is defined as the straight line connecting point to point and is the number of dominant points in this solution as generated by the swarm algorithm. The error is computed as the sum of squared perpendicular distance from every point along the arc to the line . The fitness is computed using the following equation

(7)

where is the number of points in the stroke, is the number of points in the solution that were previously labeled as possible dominant points (), is the computed error and is the error threshold. It should be noticed that when the error is larger than the threshold the fitness is given a value to lower the fitness value of the solution. Otherwise, the system favors the least number of vertices.

**4.4.2** ***Hybrid Fitting Algorithm (AlgS2)***

The *(AlgS2)* algorithm has the same problem formulation as *(AlgS1)*. However, it has different fitness and error functions. An attempt is made to fit each segment into a line or circular arc. The errors of both circle and line fit estimations are computed for each segment . The approximation with the lowest error value is chosen as the final approximation of this segment [6]. The sum of the approximation errors of all segments is considered the total error of the particle. The total error of the particle is computed by equation

(8)

where is the number of segments in the solution as generated by the swarm algorithm, is the minimum approximation error of curve and line approximations where is the circle approximation error and is the line approximation error as computed by [6]. The fitness is computed by the equation

(9)

where is the error and is a parameter tweaked to get minimum number of segments. The larger is, the higher the effect the number of segments will have. For our system, is selected to be 0.5 [6].

After each iteration of the swarm algorithm (*AlgS1* and *AlgS2*), each particle is refined using the following procedures:

1. For each particle each dominant point is checked to find if it was labeled before as a *possible dominant point* (computed as in section 4.1). If it was not labeled, the point is moved to the nearest labeled point. This ensures that all of the points generated by the DPSO are possible dominant points .

2. The particles are tested to make sure that the distance between every two successive dominate point is larger than , where is 5% of the total length of the stroke. Otherwise, two temperary particles are evaluated by removing either point from the current particle. After that, one of these points is removed based on the error caused by its removal.

3. Each two successive segments are tested to determine if they can be merged into a single segment. If merging the two segments results in segmentation that has an error value less than or equal to the error before merging, the two segments are merged. This step ensures that the final segmentation of the system has minimum number of segments.

**4.5** **Recognition**

After the user draws all strokes of the symbol, the set of un-recognized strokes is grouped together along with their segmentation as input to the feature extraction process. A composite set of features is used to generate a single feature vector. The following list describes each feature in details.

**4.5.1** **Structural and geometrical Features (FS1)**

They are features defines the structure of the geometrical symbol such as;

• *Segments:* The number of segments in the symbol.

• *Strokes:* The number of strokes or partial strokes that created the symbols.

• *Primitives:* The number of primitives in the symbol. This feature helps when identifying symbols with mixed geometric primitives such as cylinders and callouts.

• *Curves:* The number of curves or ellipses in the symbol.

• *Lines:* The number of lines in the symbol.

• *Perpendicularlines:* The number of perpendicular lines.

• *Parallellines:* The number of parallel lines.

• *Intersections:* The number of intersection between lines and curves.

• *T intersections:* The number of T intersections.

• *L intersections:* The number of L intersections.

• *X intersections:* The number of X intersections.

**4.5.2** **Rubine Features (FS2)**

These features introduced by Rubine [20] for single stroke gestures. Such as length of stroke, cosine of starting angle, smoothness of symbol and the properties of the bounding box of the symbol.

**4.5.3** **Statistical Features (FS3)**

These features use moments and statistical properties such as;

• *Zernike moments:* The zernike moments are orthogonal moments which define the distribution of point in the input space. They are invariant to both reflection and rotation of the input shape [4].

**4.5.4** **Composite Features (FS4)**

These features are composites of different geometrical and statistical properties of the symbol.

• *Size Ratio:* The ratio between width and height of the symbol.

• *Ink density:* The density of points inside its bounding box[11].

• *Convex Hull Area:* The area of convex hull with respect to area of bounding box of symbol.

• *Convex Hull Perimeter:* The perimeter of convex hull with respect to total length of symbol.

• *Mean Centroidal radius:* The Mean of the centroidal radius is the distance from each point in the symbol to the center of gravity.

• *Mean Time difference:* The mean of the time difference between each two successive points in the symbol.

After computing the features, the symbol is introduced to the classifier as a feature vector. The system uses Support Vector Machine (SVM) classifier with Linear kernel [5]. An OVO classifier structure (object versus object) is used to handle multi-class problem.

**5** **Experimental**

In our experiment we used three different datasets. The first dataset (Figure 6) was collected by Hse and Newton[12] (Hs-DS). The data are drawn by 16 users each of them was asked to draw 13 shapes from 30 to 50 times. Figure 6 and 6 shows the symbols we manually gathered from 7 different users in electrical (EL-DS) and logic design (LD-DS) domain respectively. Table 1 shows a comparison between the three datasets used in our experiments. The datasets were divided into training and test sets. Five different splits for the training and test data are generated from each dataset. The results displayed are the average recognition accuracy of the five splits. The accuracy is computed as the number of correctly recognized samples divided by the total number of samples in the test.

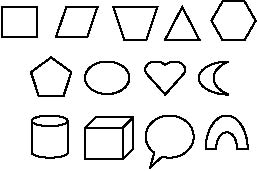
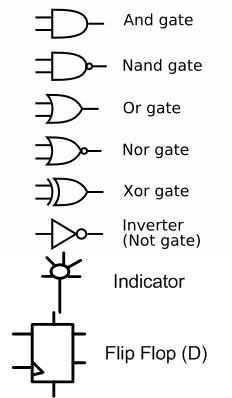


Figure 6: The Hs-DS Symbol Set

The dataset consist of simple presentation symbols.

[The Digital Design Symbols Dataset(LD-DS)]



[The Electrical Symbols Dataset(EL-DS)]

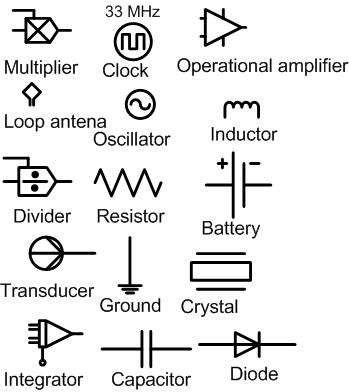


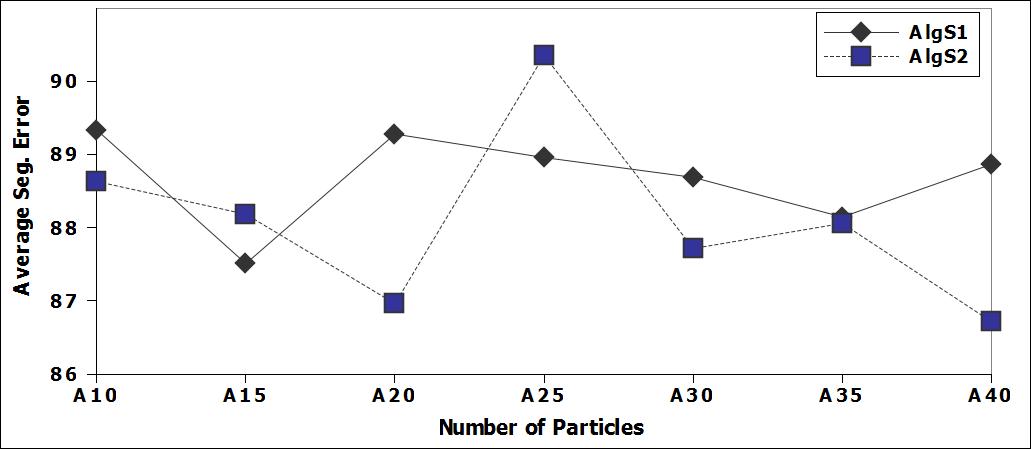
Figure 7: The datasets that were manually collected

a) The (LD-DS) dataset consist of symbols that are visually similar (i.e. has low shape diversity). b) The (EL-DS) dataset consist of complex topological symbols.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Hs-DS | EL-DS | LD-DS |
| No of samples | 7791 | 2764 | 1859 |
| No of categories | 13 | 15 | 8 |
| Avg. Samples per Category | 600 | 184 | 232 |
| No of users | 16 | 7 | 7 |
| Balanced | Yes | No | No |
| Splits | 5 random splits | 5 random splits | 5 random splits |
| Examples | Figure 6 | Figure 6 | Figure 6 |

Table 1: Table compares between datasets used in experiments in terms of number of samples, number of samples per category, etc...

A set of tests were performed on the system to enhance non-ellipse fitting algorithms. The effect of the number of particles, the error thresholds and various other parameters were investigated. The results of these tests are shown in Figures 7 and 7. Figure 7 shows that as the number of agents increases the error value decrease (error of the segmentation algorithm). The final number of particles and maximum swarm iterations used in the system was based on a tradeoff between error achieved versus the computation time. The effect of other algorithm parameters such as , (mentioned in Section3) is tested using similar tests (Figure 7, 7).

[Segmentation Error Vs. Number of particles] 

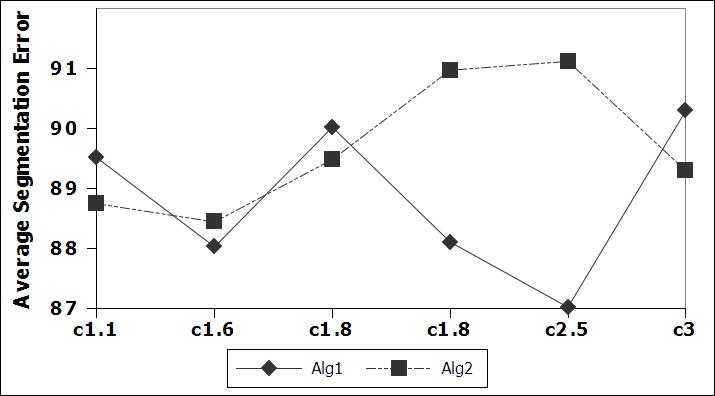
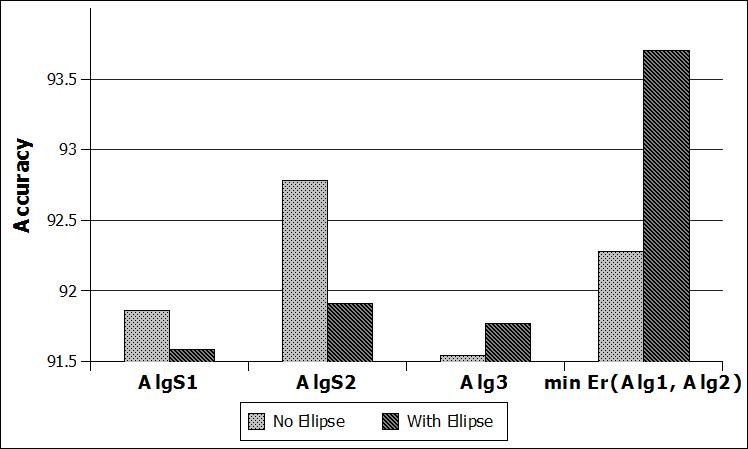
[Segmentation Error Vs. PSO parameters.] 

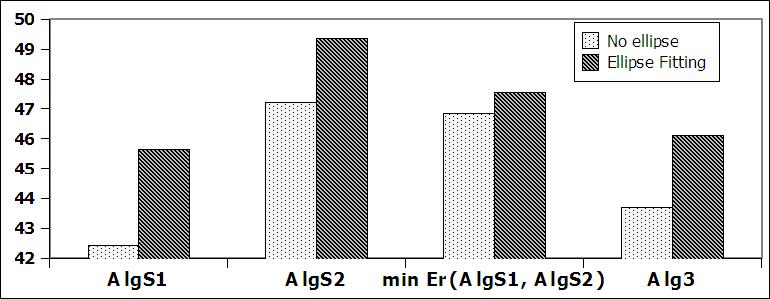
Figure 8:

a) The error decrease as the number of particle increase. b) The behavior of segmentation error versus c1.

**5.1** **Algorithms comparisons**

We performed several experiments to evaluate the presented recognition system. Firstly, we tested recognition accuracy of shapes in the data set with both algorithms (*AlgS1* and *AlgS2*). We also implemented the segmentation algorithm described in [21] (*Alg3*) to use as reference to compare it with our swarm algorithms. Figure 8 and 8 shows the accuracy achieved by each algorithm. The two swarm algorithms were tested with and without the ellipse fitting module. Combining the ellipse detection module improves the performance. The result shows that (*AlgS2*) gives better performance than Alg3 and AlgS1 in all datasets. The results also shows that combining (*AlgS1* and *AlgS2* using minimum error) outperforms AlgS2 in Hs-Ds only. This behavior is explained by noticing that features are computed based on the chosen segmentation. This means that different segmentation for a symbol generates different set of features used for recognition. This behavior affect Ls-Ds and EL-Ds more than Hs-Ds due to the higher topological and structure symbols in both datasets than of Hs-Ds.

[Algorithm comparison on HS-DS] 

[Algorithm comparison on LD-DS] 

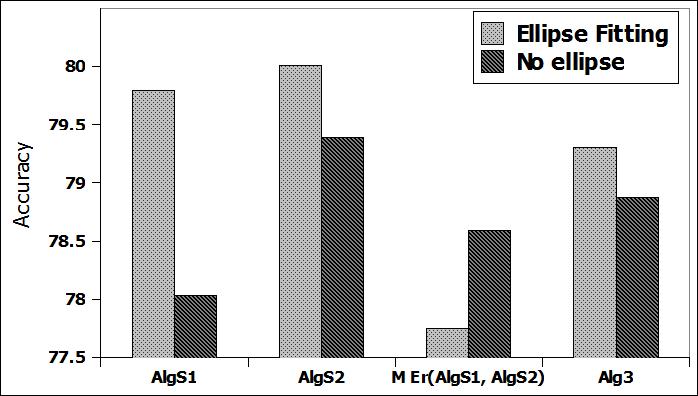
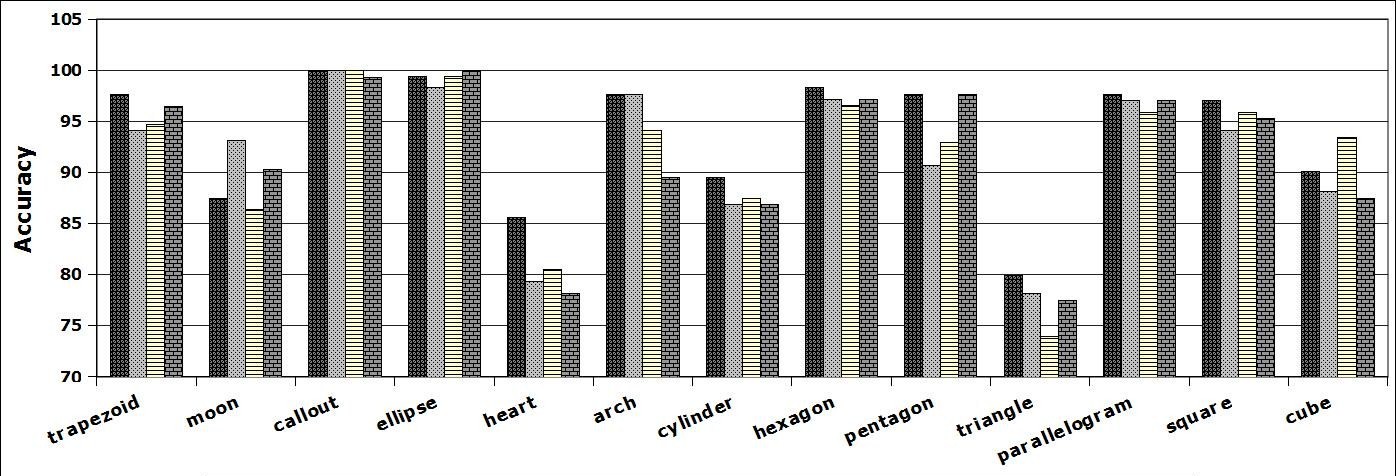
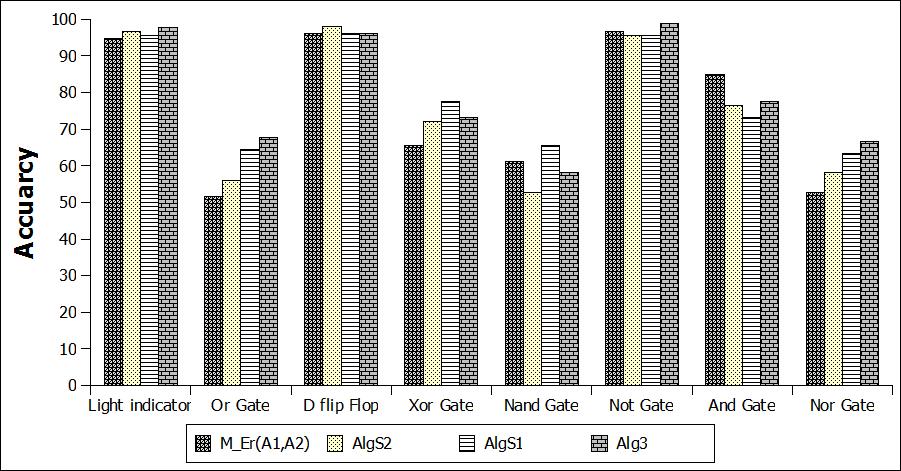
[Algorithm comparison on EL-DS] 

Figure 9: The recognition rate of different algorithms.

**5.2** **Shape Complexity Analysis**

[Symbols Comparison on HS-DS] 

[Symbols Comparison on LD-DS symbols] 

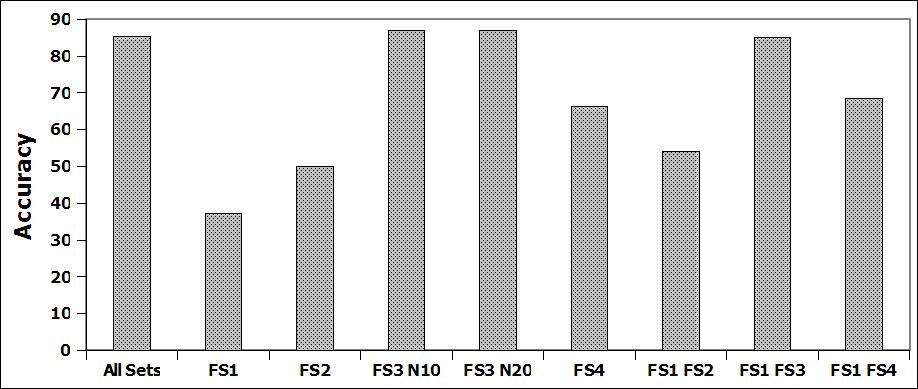
Symbol Comparison

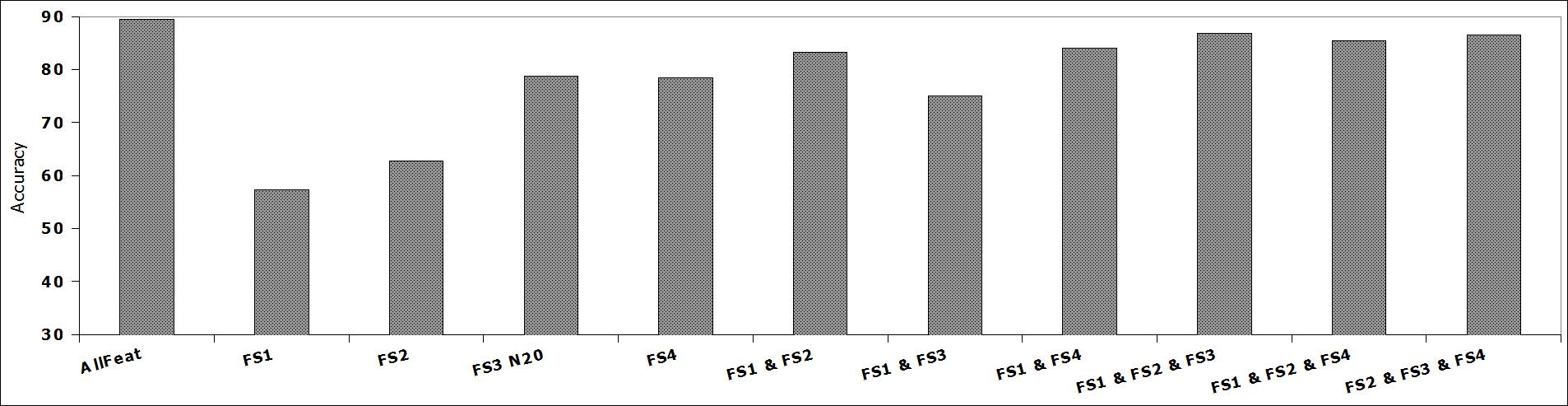
The accuracy of achieved by each algorithm in each symbol. a) Low accuracy level appears in triangle, heart, and moon symbols. b) Low accuracy level appears only in symbols that are visually confusing (i.e Nor, OR, Nand gates). The second experiment we implemented was to investigate the effect of symbol complexity and its type on the recognition rate. Figure 5.2 and 5.2 show the achieved accuracy of each symbol by our algorithms compared to (*Alg3*) [21]. It is clearly noted that symbols that have only line segments achieve higher accuracy rate than other symbols (such as square, parallegram). The results also indicate that algorithm *AlgS1* achieves better performance than algorithm *AlgS2* in the symbols that consists only of lines. This is understandable because algorithm *AlgS1* divides strokes into line segments only but *AlgS2* is able to divide strokes into lines and curves based on the minimum error of the segment itself. Figure 5.2 shows that in logic design (LD-DS) *AlgS2* achieved better performance than *Alg3* and *AlgS1*. This proves that for complex shapes that are combination of lines and arcs are *AlgS2* achieves better segmentation. Algorithm *Alg3* gives good performance as long as the symbols consist of combination of lines and curves, if the stroke consists of curve or lines only the algorithm may lead to wrong segmentation result. This is because the system divides the stroke first into line segments then tries to decide whether each segment can be represented better as a curve. However, in *AlgS2* the curve segments are tested while choosing the best segmentation. Combining both algorithm *AlgS1* and *AlgS2* improved the recognition rate of all symbols. The penalty for this improved performance is the computational time required to run both swarm algorithms. Table 2 shows the confusion matrix for the (HS-DS) symbols. The table shows that an error occurs mostly between two sets of symbols: moon with triangles and trapezoid with triangles. This observation is understandable as the symbols are visually similar but the system must be able to differentiate between them. This leads us to the next experiment to choose the best set of feature to recognize symbols.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Categories | ellipse | heart | trapezoid | pentagon | arch | hexagon | square | triangle | cube | cylinder | parallelogram | moon | callout |
| ellipse | 150 | 20 | 0 | 0 | 0 | 5 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| heart | 2 | 130 | 0 | 1 | 10 | 1 | 1 | 0 | 0 | 0 | 1 | 4 | 0 |
| trapezoid | 0 | 1 | 147 | 5 | 1 | 0 | 1 | 3 | 2 | 0 | 37 | 0 | 0 |
| pentagon | 0 | 0 | 1 | 139 | 0 | 7 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| arch | 0 | 0 | 1 | 1 | 125 | 0 | 4 | 0 | 3 | 0 | 0 | 15 | 1 |
| hexagon | 0 | 0 | 0 | 2 | 0 | 135 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| square | 0 | 0 | 0 | 0 | 0 | 0 | 138 | 0 | 0 | 0 | 0 | 1 | 0 |
| triangle | 0 | 0 | 1 | 2 | 5 | 0 | 1 | 148 | 0 | 0 | 4 | 7 | 0 |
| cube | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 146 | 5 | 0 | 0 | 0 |
| cylinder | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 147 | 0 | 0 | 0 |
| parallelogram | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 2 | 0 | 107 | 0 | 0 |
| moon | 0 | 1 | 0 | 0 | 10 | 2 | 3 | 0 | 0 | 0 | 0 | 125 | 5 |
| callout | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 145 |

Table 2: Confusion Matrix of Hs-DS

**5.3** **Features Analysis**

Different feature sets are tested to determine the best features that can be used in sketch and symbol recognition. Figure5.3 shows the result of using different feature sets (from the basic four sets **FS1,FS2,FS3,FS4** in Section 4.5). Results show that geometerical features (**FS1**) and Rubine features (**FS2**) [20] gives the low results when used alone without any other features. This is because they are mainly computed for single stroke gestures and fare bad in multi-stroke symbols [24]. Features **(FS4)** gives good results and it is further improved by adding geometerical structural features **(FS1)**. [Feature Comparison on Hs-DS] 

[Feature Comparison on EL-DS] 

Different Features

The effect of different feature set on accuracy. FS4 and FS3 give good results in a) Hs-DS and b) EL-Ds. The results are improved by combining FS1 with FS2 and FS1 with FS4. We could not test the result of the segmentation algorithm alone due to the fact that the correct segmentation is highly qualitative. It is only based on what the user intends to draw while sketching the symbol. The main features affected by the segmentation stage are the geometric features computed in **FS1**. Hence those are the features we used to estimate the efficiency of the segmentation algorithm. The recognition accuracy using only the geometrical features is reported in Figure 10. The figure shows that *AlgS2* achieve better results.

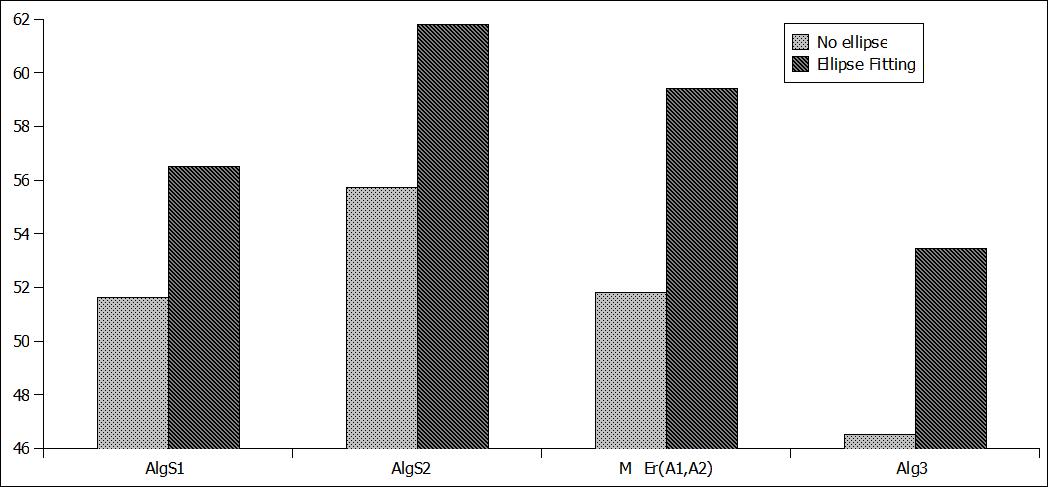
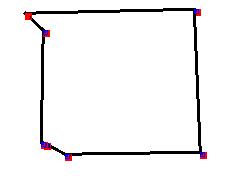
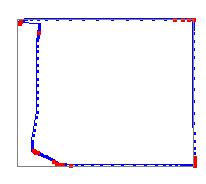


Figure 10: The effect of segmentation on Hs-DS accuracy

[User strokes] [Segmented strokes]



[Multiple symbols]

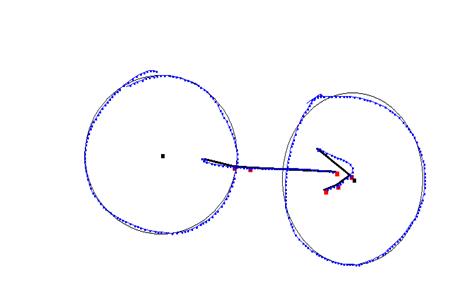


Figure 11: Segmentation outputs

The output segmentation of different input strokes. a) Sample of single input stroke b) The single stroke after segmentation. c) An example of multiple strokes and their segmentation.

**6** **Conclusion and Future Work**

This paper presented a new approach to sketch recognition using DPSO. The system uses multiple preliminary information, i.e., speed and curvature information, as input to the *DPSO*. It is noted that the *DPSO* in general generates an optimized stroke segmentation which improves the final recognition rate. The tradeoff between accuracy achieved and time complexity can be further investigated to achieve better results. The final recognition is based on both global shape and hierarchical stroke properties. Adding stroke and segment geometrical features improved the final recognition. Global shape and moment properties were used to prevent the need to use graph matching which is computationally expensive [2]. The experiments evaluated two*DPSO* different segmentation algorithms (AlgS1 and AlgS2) on three different datasets (Hs-Ds, EL-Ds and LD-Ds). Results show that AlgS2 achieves better segmentation and final accuracy than Alg3 [21] on the datasets used. The results proved that the system is easily expandable to more and more symbols as it does not depend on low level primitive recognizers but on high level symbol and geometrical features. The system achieved an overall improvement of more than 2% over Alg3 in all datasets. A possible extension of this research is to complete the clustering algorithm for fully automated sketch recognition. Currently, the system processes one symbol at a time, an enhancement would be to allow users to draw more than one symbol at a time and incorporate a method for separating symbols. Another area of enhancements is the features extraction methods. Introducing more spatial and geometrical features is believed to improve classifications. Features that represents the appearance of the symbols are likely to improve the recognition that have similar geometrical structure (i.e. Nor and OR gate in logic symbols).

**References**

[1] Peter Agar and Kevin Novins. Polygon recognition in sketch-based interfaces with immediate and continuous feedback.  *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 147--150, 2003. ACM Press.

[2] Christine Alvarado and Randall Davis. SketchREAD: a multi-domain sketch recognition engine.  *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 34, 2007. ACM Press.

[3] Alvarado, Christine and Lazzareschi, Michael. Properties of Real-World Digital Logic Diagrams.  *PLT '07: Proceedings of the First International Workshop on Pen-Based Learning Technologies*, 2007. IEEE Computer Society.

[4] Bailey,, Robert R. and Srinath,, Mandyam. Orthogonal Moment Features for Use With Parametric and Non-Parametric Classifiers.  *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):389--399, 1996.

[5] Chih-Chung Chang and Chih-Jen Lin.  *LIBSVM: a library for support vector machines*. 2001.

[6] Ke-Zhang Chen and Xi-Wen Zhang and Zong-Ying Ou and Xin-An Feng. Recognition of digital curves scanned from paper drawings using genetic algorithms.  *Pattern Recognition*, 36(1):123-130, 2003.

[7] N. Chernov and C. Lesort. Least squares fitting of circles and lines.  *CoRR*, cs.CV/0301001, 2003.

[8] M. Fonseca and C. Pimentel and J. Jorge. Cali-an online scribble recognizer for calligraphic interfaces. 2002.

[9] Walter Gander and Gene H. Golub and Rolf Strebel. Least square Fitting of Circles and Ellipses.  *??*, 1994.

[10] Leslie Gennari and Levent Burak Kara and Thomas F. Stahovich and Kenji Shimada. Combining geometry and domain knowledge to interpret hand-drawn diagrams.  *Computers & Graphics*, 29(4):547--562, 2005.

[11] Leslie Gennari and Levent Burak Kara and Thomas F. Stahovich and Kenji Shimada. Combining geometry and domain knowledge to interpret hand-drawn diagrams.29(4):547-562, 2005.

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

[13] Levent Burak Kara and Thomas F. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches.  *Computers & Graphics*, 29(4):501--517, 2005.

[14] Levent Burak Kara and Thomas F. Stahovich. Hierarchical parsing and recognition of hand-sketched diagrams.  *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 13--22, 2004. ACM Press.

[15] Kennedy, J. and Eberhart, R. Particle Swarm Optimization. pages 1942--1948 vol.4, 1995.

[16] Kennedy, J. and Eberhart, R. C. A discrete binary version of the particle swarm algorithm.  *Systems, Man, and Cybernetics, 1997.'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on*, 5, 1997.

[17] Oltmans, Michael.  *Envisioning sketch recognition: a local feature based approach to recognizing informal sketches*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2007.

[18] Beryl Plimmer and Mark Apperley. Computer-aided sketching to capture preliminary design.  *CRPITS '02: Third Australasian conference on User interfaces*, pages 9--12, Darlinghurst, Australia, Australia, 2002. Australian Computer Society, Inc.

[19] Khaled S. Refaat and Wael N. Helmy and AbdelRahman H. Ali and Mohamed S. AbdelGhany and Amir F. Atiya. A new approach for context-independent handwritten offline diagram recognition using support vector machines. *International Joint Conference on Neural Networks, IJCNN*, pages 177-182, 2008. IEEE.

[20] Dean Rubine. Specifying gestures by example.  *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 329--337, 1991. ACM Press.

[21] Tevfik Metin Sezgin and Thomas Stahovich and Randall Davis. Sketch based interfaces: early processing for sketch understanding.  *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1--8, 2001. ACM Press.

[22] Peng-Yeng Yin. A discrete particle swarm algorithm for optimal polygonal approximation of digital curves.  *J. Visual Communication and Image Representation*, 15(2):241-260, 2004.

[23] Bo Yu. Recognition of freehand sketches using mean shift.  *IUI 03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 204--210, 2003. ACM Press.

[24] Lisha Zhang and Zhengxing Sun. An experimental comparison of machine learning for adaptive sketch recognition.  *Applied Mathematics and Computation*, 185(2):1138--1148, 2007.

1. Alg3 used in Section 5 [↑](#footnote-ref-2)
2. By trial and error best threshold found was [↑](#footnote-ref-3)