Using Fuzzy Logic to Recognize Geometric Shapes Interactively

Manuel J. Fonseca Joaquim A. Jorge
Departamento de Engenharia Informática, IST/UTL
Av. Rovisco Pais, 1049-001 Lisboa, PORTUGAL
mjf@ist.utl.pt, jorgej@acm.org

Abstract–**This paper presents a simple method, based on fuzzy logic, to recognize multi-stroke sketches of geometric shapes and uni-stroke gestural commands. It uses temporal adjacency and global geometric properties of figures to recognize a simple vocabulary of geometric shapes drawn in different line styles. The geometric features used (convex hull, largest-area inscribed and smallest-area enclosing polygons, perimeter and area ratios) are invariant with rotation and scale of figures. Through experimental evaluation we have found the method very usable with acceptable recognition rates although the multi-stroke approach poses problems in choosing appropriate values for timeouts.**

Although we have privileged simplicity over robustness, the method has proved suitable for interactive applications.

Keywords– **Symbol Recognition, Fuzzy Logic, Tools and Techniques, Multi-stroke Shapes**

I. Introduction

Recognition of hand-drawn geometric shapes has garnered new attention with the widespread adoption of Personal Digital Assistants (PDAs) [10], [9]. While conventional off-line approaches to recognition of geometric figures have long focused on raw classification performance, on-line systems raise a different set of issues. First, geometric figures in CAD drawings are input precisely either by human drafts-people or through computer peripherals. Recognizing these shapes deals mainly with noise introduced by sources outside the process, such as copy degradation or poor photographic reproduction. In online applications however, the noise is inherent to the process of information gathering and shapes are often sketched poorly due to media, operator and process limitations, yielding imperfect and ambiguous shapes that even humans find difficult to distinguish. One further difference from off-line algorithms is that input data are sequences of points rather than image bitmaps. Also, the online recognizer should cope with innumerable variations in hand sketches — it is not reasonable to require that all geometric shapes be drawn in a single stroke, especially if we want to recognize dashed lines.

Our approach extends previous work by Kimura [7] who built a simple multi-stroke recognizer for axis-aligned (i.e. non-rotated) shapes using geometric features alone. We extend his work by recognizing arbitrarily rotated shapes, drawn in different line styles (dashed, bold) and adding single stroke gestures such as delete and undo, which are commonly used in CAD systems. Most notably we extend this work, by using fuzzy logic to handle imprecision and ambiguity in hand drawn sketches in a natural manner.

We want to emphasize that our main concern is to be able to recognize positive samples, i.e. shapes the user intended to draw in the first place. As such, our motivation was not to develop a foolproof method, but rather to provide a tool to aid users in constructing diagrams and other structured drawings interactively. In the remainder of this paper we describe the feature extraction and classification processes, discuss experimental results and future work.

II. The Recognition Algorithm

The recognition method is based on three main ideas. First, it uses entirely global geometric properties extracted from input shapes. Since we are mainly interested in identifying geometric entities, the recognizer relies mainly on geometry information. Second, to enhance recognition performance, we use a set of filters either to identify shapes or to filter out unwanted shapes using distinctive criteria. Third, to overcome uncertainty and imprecision in shape sketches, we use fuzzy logic [2] to associate degrees of certainty to recognized shapes, thereby handling ambiguities naturally.

This algorithm recognizes elementary geometric shapes, such as `Triangles`, `Rectangles`, `Diamonds`, `Circles`, `Ellipses`, `Lines` and `Arrows`, and five gestural commands, `Delete`, `Undo`, `WavyLine`, `Move` and `Copy`, as depicted in Figs. 1 and 2. Shapes are recognized independently of changes in rotation, size or number of individual strokes. Commands are shapes drawn using a single stroke. The recognizer works by looking up values of specific features in fuzzy sets associated to each shape and command. This process yields a list of plausible shapes ordered by degree of certainty.

This approach extends and improves Kimura's work [7],

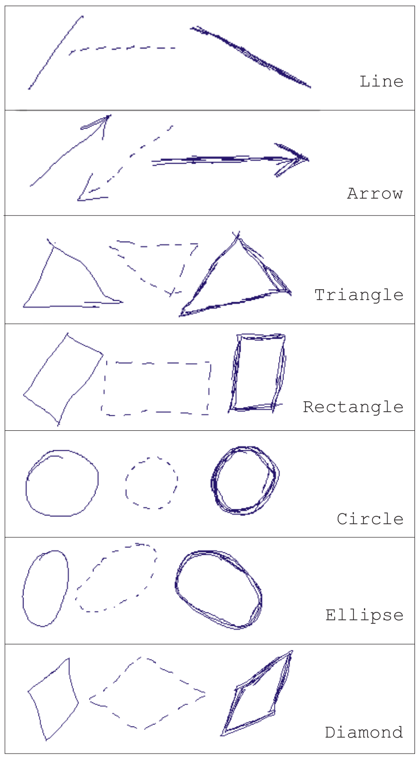

Fig. 1. Multi-stroke geometric shapes.

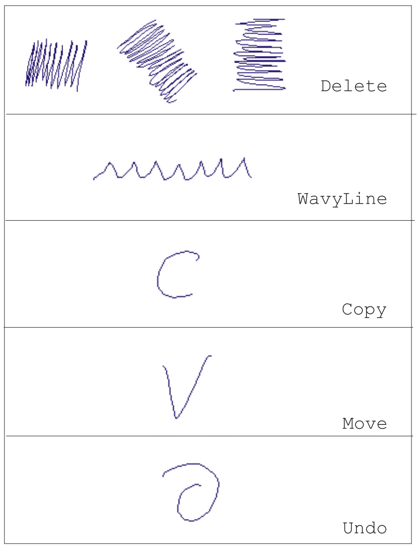

Fig. 2. Uni-stroke shapes.

which did not distinguish rotated, open / closed, dashed and bold shapes. The present is an evolution of previous work [6], which did recognize less shapes and used a decision tree to prune out incorrect classifications. We found out that using classification rules alone, provides more flexibility and extensibility without sacrificing robustness. Other authors have proposed more complex methods, involving neural networks [4] using a procedure that might prove more robust than ours, although such claims are not made explicit and substantiated by experimental results.

III. Geometric Feature Selection

We start the recognition process by collecting data points using a digitizing tablet, from the first pen-down event until a set timeout value after the last pen-up. Next, we compute the convex hull of the set of input points thus collected, using Graham's algorithm [8]. We then use this convex hull to compute three special polygons. Using a simple three-point algorithm we identify the largest-area triangle that fits inside the convex hull. The second polygon is the largest-area inscribed quadrilateral [3] and the third is the smallest area enclosing rectangle [5] (See Fig. 3). Finally, we compute the area and perimeter of each polygon to estimate features and degrees of likelihood for each shape class.

A *Stroke* is defined as the set of points from pen-down to pen-up. A *Scribble* is a sequence of strokes collected within the timeout value. A *Shape* is a recognized scribble. It relates the input points drawn to a class and a set of geometric attributes, such as, start- and end-points, bounding box.

The set of shapes selected and presented in Figs. 1 and 2 are the basic elements to allow the construction of technical diagrams such as electric or logic circuits, flowcharts or architectural sketches. These types of diagrams also require distinguishing between solid, dash and bold depictions of shapes in the same family. Typically, Architects will use multiple overlapping strokes to embolden lines in sketches. This mechanism is commonly used in drawing packages. We are therefore interested in recognizing different renderings of a given shape as illustrated in Fig. 1. We are just interested in collecting qualitative differences, not in obtaining precise values for attributes, without regard to different linestyles and widths.

In order to select the features that best identify a given shape, we built percentile graphics for each feature. These graphics illustrate the statistical distribution of feature values over the different classes, extracted from sample drawings. For each shape, the solid bar spans the 25% to 75% percentiles, while the line extends from 10% to 90% of all observed values of a given feature.

Our initial selection of features takes into account specific properties of shapes to identify. Associated with these features we infer fuzzy sets from training data, which express the allowable values for a given shape. Usually one feature alone is not enough to distinguish shapes, yielding incorrect classifications. We then add extra features (with corresponding fuzzy sets) to weed-out unwanted classifications, yielding more complex rules as needed.

To distinguish `Circles` from other shapes we use the *Thinness* ratio (P_{ch}^2/A_{ch}), where A_{ch} is the area of the convex hull and P_{ch}^2 is its perimeter squared. The thinness of a circle is minimal, since it is the planar figure with smallest perimeter enclosing a given area, yielding a value near 4π (see Fig. 4). In this figure the thinness values for `Lines`, `Arrows` and `WavyLines` lie outside the range of values indicated. We chose not to indicate these values in order to make the range of values for `Circles` more visible.

We identify `Lines` using another thinness ratio, which compares the height of the (non-aligned) enclosing rectangle (H_{er}) with its width (W_{er}). The H_{er}/W_{er} ratio will have values near zero for lines and bigger values for other shapes (see Fig. 5). This feature is more reliable for distinguishing bold lines than P_{ch}^2.

In order to identify `Rectangles` we use two ratios. One relates the convex hull to enclosing rectangle while the other measures the largest quadrilateral that fits inside the convex hull against the enclosing rectangle. For rectangular shapes

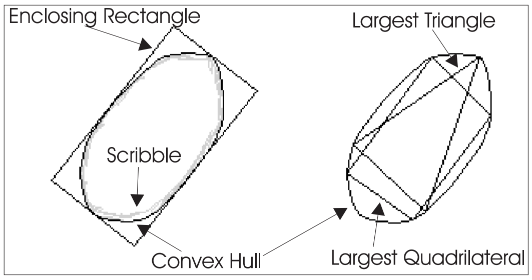

Fig. 3. Polygons used to estimate features.

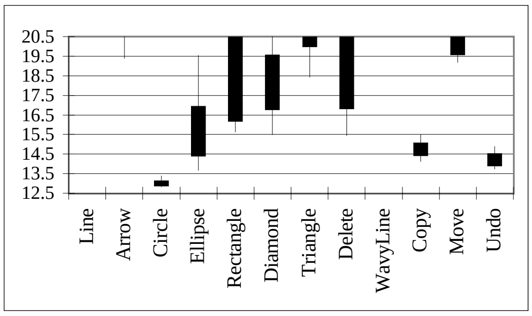

Fig. 4. Percentiles for the P_{ch}^2/A_{ch} ratio.

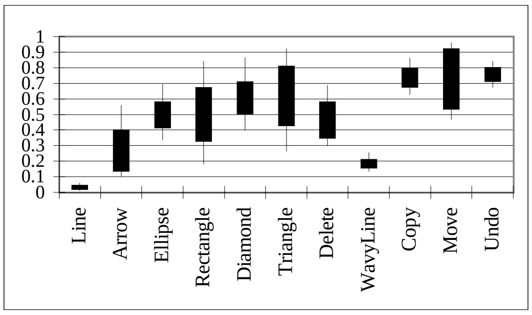

Fig. 5. Percentiles for the H_{er}/W_{er} ratio.

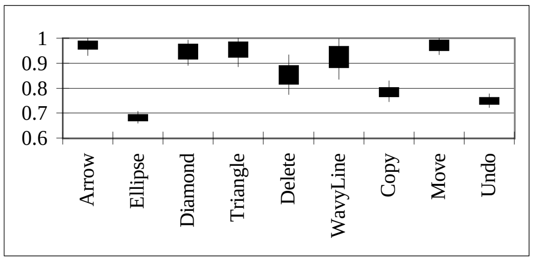

Fig. 7. Percentiles for the A_{lq}/A_{ch} ratio.

the area of the convex hull will be very close to that of the enclosing rectangle (A_{er}) and this one will be very close to the largest quadrilateral's (A_{lq}). The A_{ch}/A_{er} and A_{lq}/A_{er} ratios will have values near unity for rectangles (see Fig. 6).

We identify `Ellipses` by comparing the area of the largest quadrilateral to that of the convex hull. The A_{lq}/A_{ch} ratio will have values near 0.7 for ellipses and bigger values for other shapes (see Fig. 7). This is more robust than using A_{ch}/A_{er} which is too ambiguous.

Since this recognizer is intended for interactive use, we are interested in detecting gestures such as a set of zigzag strokes drawn in succession to signify erasing objects underneath. Using our geometry approach, we detect this pattern by comparing the total length of strokes (T_l) drawn to the perimeter of the convex hull (P_{ch}). The T_l/P_{ch} ratio has large values for the `Delete` command, and smaller values for other shapes (see Fig. 8).

To distinguish `Diamonds` from other shapes we divide the area of the largest triangle that fits inside the convex hull (A_{lt}) by the area of the largest quadrilateral. The A_{lt}/A_{lq} ratio has values between 0.5 and 0.6 for diamonds and bigger ones for other shapes (see Fig. 9).

To identify `Triangles` and `Move` gestures we compare the area and perimeter of the largest triangle to that of the convex hull. A_{lt}/A_{ch} and P_{lt}/P_{ch} will have values near unity for triangles and moves, and smaller values for other shapes (see Fig. 10). We distinguish these two shapes using the openness of the move command.

The `Copy` and `Undo` commands are recognized using a combination of two fuzzy sets. First we separate them from others using the thinness ratio P_{ch}^2/A_{ch} and then we separate one from the other using the T_l/P_{ch} ratio (see Fig. 11).

The horizontal `WavyLine` gesture has two attributes that allow the distinction from other shapes. First it is a little "fatter" than lines, so the H_{er}/W_{er} ratio has bigger values. Second, it is drawn always the same way (that is, X values are always increasing). To measure that we compare the width of the bounding box (W_{bb}) with the absolute horizontal movement (H_m), thus the W_{bb}/H_m ratio has values near unity for wavyLines (see Fig. 12).

We have found `Arrows` the hardest to identify, because we have not found a feature that characterizes them well. Arrows are identified by local features while our recognizer deals with global features, such as the convex hull area or the enclosing rectangle perimeter. We consider this a limitation of our approach.

To distinguish `Dashed` from solid lines we use the ratio $P_{ch}{*}N_s/T_l$ where N_s is the number of strokes in the scribble. Because dashed shapes have a large number of strokes, with

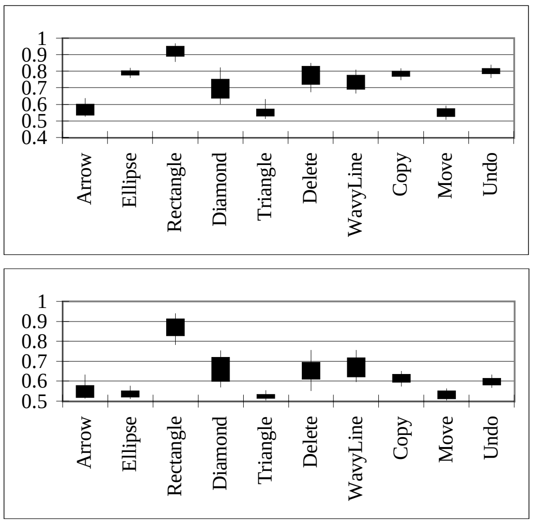

Fig. 6. Percentiles for A_{ch}/A_{er} (top) and A_{lq}/A_{er} (bottom) ratios.

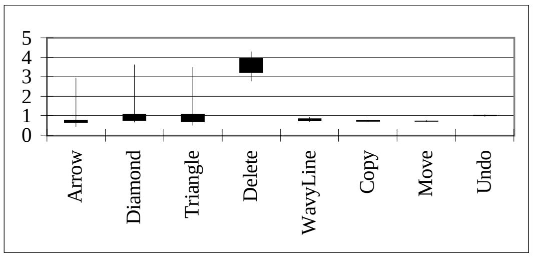

Fig. 8. Percentiles for the T_l/P_{ch} ratio.

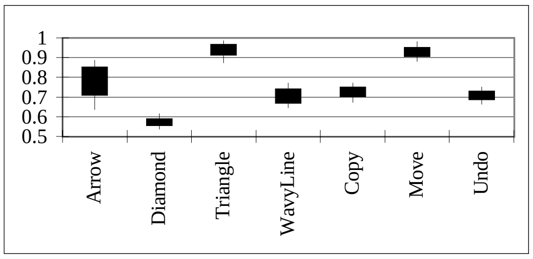

Fig. 9. Percentiles for the A_{lt}/A_{lq} ratio.

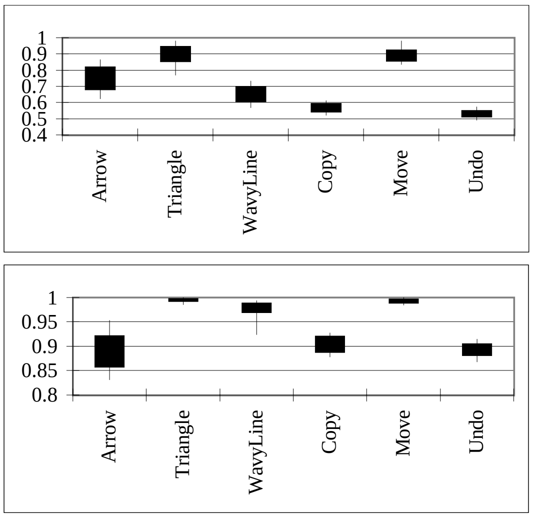

Fig. 10. Percentiles for A_{lt}/A_{ch} (top) and P_{lt}/P_{ch} (bottom) ratios.

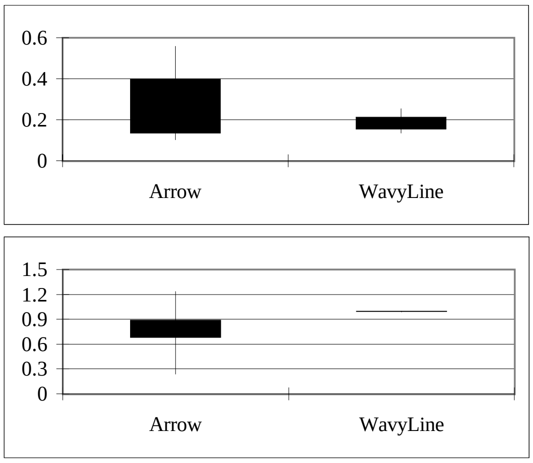

Fig. 12. Percentiles for H_{er}/W_{er} (top) and W_{bb}/H_m (bottom) ratios.

a small total length, this ratio exhibits large values for dashed shapes and smaller values for closed shapes.

The T_l/P_{ch} ratio separates `Bold` from solid lines. Values bigger than 1.5 indicate redundant strokes (i.e. Bold lines).

Recognizing shapes drawn using overlapping strokes to signify `bold` linestyle conflicts with the `Delete` command, because we use the same feature (T_l/P_{ch}) in the same manner to identify both. To solve this conflict we introduce a heuristic feature to distinguish “filled” from “hollow” shapes. Hollow shapes do not have points inside near the barycenter. To compute *hollowness* we first calculate a triangle, whose dimensions are roughly 60% of the largest triangle that fits inside the convex hull and shares the same barycenter. We then count scribble points lying inside the smaller triangle. Shapes like `Triangles`, `Rectangles`, `Circles`, `Ellipses` or `Diamonds` shouldn't have any point inside, but `Arrows`, `WavyLines` or `Delete` gestures must have (see Fig. 13).

Fig. 14 lists all shapes identified by the recognizer and the features used by each. As described earlier, conductive rules combine assertions or negations of these features as we shall see in the next section.

IV. The Fuzzy Recognition Method

The recognition method starts by collecting points from a digitizing tablet and computing some important polygons. Us-

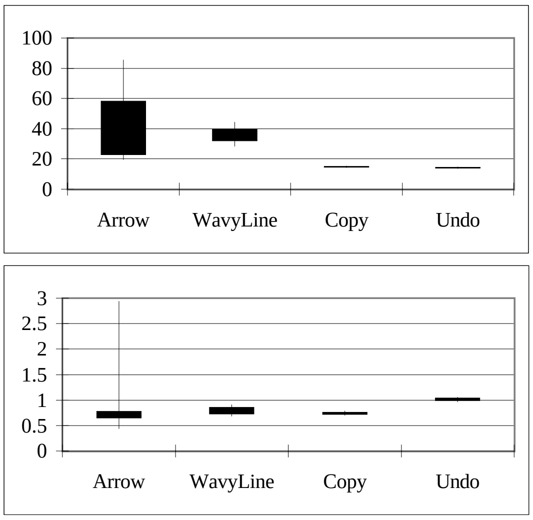

Fig. 11. Percentiles for P_{ch}^2/A_{ch} (top) and T_l/P_{ch} (bottom) ratios.

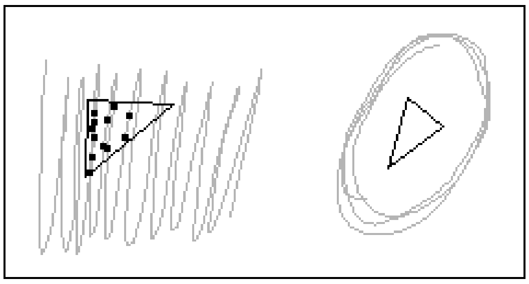

Fig. 13. Hollowness of `Delete` and `Ellipse`.

	Hollowness	Pch2/Ach	Her/Wer	Tl/Pch	Ach/Aer	Alq/Ach	Plt/Pch	Hm/Wbb	Alt/Ach	Alq/Aer	Plq/Pch	Alt/Alq	Pch*Ns/Tl	Vm/Hbb	Alt/Aer
Line			X												
Arrow	X		X		X		X				X				
Triangle	X						X		X						
Rectangle	X				X					X					
Circle	X	X													
Ellipse	X	X				X									
Diamond	X	X				X				X		X			
Delete	X		X	X	X										
WavyLine	X		X	X				X							
Copy		X		X				X			X			X	
Move		X		X			X		X				X		
Undo	X	X		X		X									X

Fig. 14. Features used by each shape.

ing these polygons area and perimeter we compute values for features selected as described in the previous section. We use fuzzy sets associated with each feature to classify the shape class(es) for a set of strokes (scribble). These sets were derived from percentile graphics like the ones shown in Figs. 4 to 12. We identify shapes drawn by computing their degree of membership (dom) using the fuzzy sets associated with each feature. If several shapes are identified, the recognizer returns all classifications ordered by degree of certainty.

The next two subsections describe how we derive fuzzy sets from experimental data and how to obtain the final classification results.

A. *Deriving Fuzzy Sets from Training Data*

To choose the “best” fuzzy sets describing each shape class we use a “training set” developed by three subjects, who drew each shape thirty times; ten times using solid lines, ten times using dashed lines and ten times using bold lines. Based on this training set we define several ratios, combining among others the area and perimeter of these three polygons described above.

Each shape is defined by several fuzzy sets, some of them are used to identify the shape and others are used to avoid “wrong” results. We did not use the same number of features for all shapes because some of them are identified univocally with one or two features. In general we tend to avoid using “too many” fuzzy sets to characterize a shape. Each set requires careful data analysis and experimentation to find the “right values”, that is, those yielding higher recognition rates and less misrecognitions (false positives). The simplest case is that of a line, which is distinguished by thinness.

As depicted in Fig. 15 a fuzzy set is defined by four values. After the selection of the set of features for each shape, we compute these four values based on the percentiles. Values `b` and `c` correspond to the percentiles 10% and 90% respectively, and `a` and `d` to the “minimum” and “maximum” after we have identified the outliers in each distribution. The elimination of outliers minimizes confusion between different shape families. There is a tradeoff in designing fuzzy sets from statistical data. If we make `a` and `d` very wide apart we increase the recognition rate as well as the number of false positives. If we select “narrower” values we decrease the number of false classifications at the cost of a much lower recognition rate. Abe [1] discuss an automated procedure for collecting this information using activation and inhibition hyper-boxes. However, their approach is not sufficiently flexible to be used here.

A.1 Ambiguity

Taking into account the shapes identified by the recognizer, we present four special cases which can yield ambiguous results. These cases are presented in Fig. 16.

Humans solve this natural ambiguity between geometric shapes identifying more than one shape and making the final distinction based on the context or in the feedback from others.

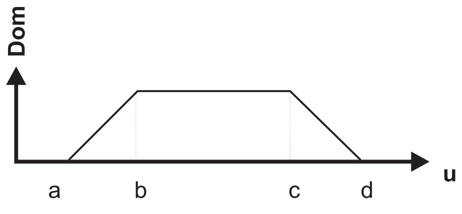

Fig. 15. Definition of a fuzzy set.

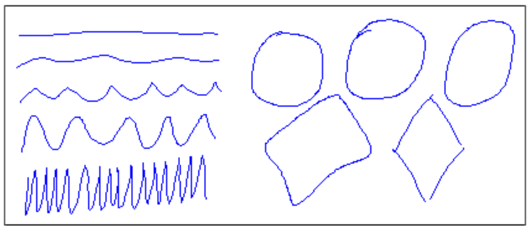

Fig. 16. Ambiguity cases among shapes.

The recognizer described in this paper deals with ambiguity between shapes in a similar way, i.e. when it can not identify univocally a geometric shape, returns a list of plausible candidates. The choice of the best candidate can then be made by the application using context information.

The ambiguities between shapes are modeled naturally using fuzzy logic to associate degrees of certainty to recognized shapes. Fig. 17 illustrates corresponding fuzzy sets for the ambiguous cases shown in Fig. 16.

B. *Deriving Results from Fuzzy Sets*

After collecting input points from the tablet and computing the special polygons from scribble data, the recognizer calculates the degree of membership for each shape class. This degree is the result of `AND` together degrees of membership for the relevant fuzzy sets as highlighted in Fig. 14.

In the following paragraphs we exemplify how to build rules that classify shapes. The first rule identifies `Lines` while the second defines `Diamonds`.

```
IF Scribble IS VERY THIN
THEN
     Shape IS Line
```

This is the simplest rule used in our recognizer. It ascertains that “very thin” scribbles should be classified as `Lines`. A more complicated rule recognizes `Diamonds`:

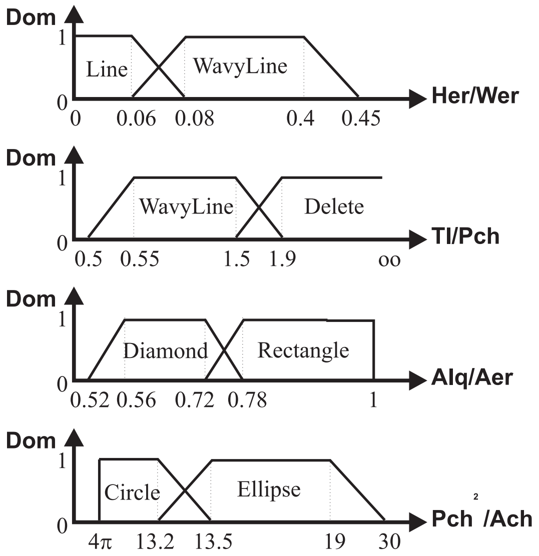

Fig. 17. Fuzzy sets representing the ambiguity cases supported by the recognizer.

```
IF $A_{lt}/A_{lq}$ IS LIKE Diamond AND
   $A_{lq}/A_{ch}$ IS NOT LIKE Ellipse AND
   $A_{lq}/A_{er}$ IS NOT LIKE Bold Line AND
   $A_{lq}/A_{er}$ IS NOT LIKE Rectangle
THEN
     Shape IS Diamond
```

Where AND denotes the conjunction of fuzzy predicates $\mu_x(f \text{AND}\, g) = \min(\mu_x(f), \mu_x(g))$ and NOT is defined by $\mu_x(\text{NOT}\, f) = 1 - \mu_x(f)$.

Fig. 9 describes the statistical distribution for feature A_{lt}/A_{lq}. Figs. 7 and 6(bottom) illustrate observed statistical distributions for the other two features respectively. We derive fuzzy sets for these features from the corresponding distributions as outlined previously.

The algorithm distinguishes between Solid, Dashed and Bold shapes after identifying "basic" shapes. This enables us to treat linestyles as attributes orthogonal to shape, making the design more modular which in turn will make it easier to recognize different shapes in the future.

V. Experimental Results

In order to evaluate the recognition algorithm, we asked several subjects to draw each shape repeatedly using solid, dashed and bold lines. Subjects were told that the experiment was meant to test recognition, so they didn't try to draw "unnatural" shapes.

We used a Wacom LCD digitizing tablet PL-300 and a cordless stylus to draw the shapes. Three subjects were experts in using pen and tablet while the others had never used a digitizing tablet. We gave a brief description of the recognizer to the users, including the set of recognizable shapes. We also told them about the multi-stroke shape recognition capabilities, the independence of changes in rotation or size and about the timeout. Novice subjects had a short practice session in order to get used to the stylus/tablet combination. During the drawing session the recognizer was turned off to avoid interfering with data collection.

The recognizer successfully identified 92% of the scribbles drawn. The overall recognition results are presented in Table I. A cursory analysis of the confusion matrix (not shown for lack space) reveals that Circles are easily confused with Ellipses, which is both an acceptable and *intuitive* behavior. If we consider Circles and "fat" Ellipses in the same shape class the recognition rate increases to 93%. Second, Diamonds are often confused with Rectangles, and have a low recognition rate. Again, this rate increases if we *consider* the two classes as the same (quadrilaterals). Using fuzzy logic again works to our advantage by allowing different degrees of likelihood to distinguish among these ambiguous classifications, diamonds being the most restrictive classification and therefore enjoying the lowest rate.

If we consider successful classification when the "correct" shape is among the top three identified, the recognition rate increases from 92% to 93%. This recognition rate further increases to 94% if we group Circles with "fat" Ellipses and Rectangles with Diamonds (cf bottom row of Table I).

VI. Discussion and Future Work

We have described a simple and fast recognizer for elementary geometric shapes. Our intent was more to provide a means to support calligraphic interaction rather than a totally robust and "foolproof" approach to reject shapes outside the domain of interest. Our experience using this recognizer in interactive settings show that the observed recognition rates make it usable for drawing input. The timeouts required to recognize shapes regardless of the number of strokes using temporal adjacency have proved to be the most unnatural constraint so far. We are working on a trainable version of this recognizer to allow us to easily add new shape classes to the core set presented in this paper. One idea is to automatically derive fuzzy sets from training data along the lines of [4] and performing principle component analysis to identify the features relevant to new shape classes.

The high recognition rates and fast response characteristic of this recognizer make it very usable in interactive applications. We are also looking at more natural input segmentation methods and improving the recognition rate and robustness of our approach through incremental computational geometry algorithms in order to make the recognizer publicly available.

TABLE I
Recognition rates.

Shape Type	First Choice	Top Three
Individual	92%	93%
Grouped	93%	94%

References

[1] Shigeo Abe and Ming-Shong Lan. Efficient methods for fuzzy rule extraction from numerical data. In C. H. Chen, editor, *Fuzzy Logic And Neural Networks Handbook*, pages 7.1–7.33. IEEE Press, 1996.

[2] James C. Bezdek and Sankar K. Pal. *Fuzzy models for pattern recognition*. IEEE Press, 1992.

[3] J. E. Boyce and D. P. Dobkin. Finding extremal polygons. *SIAM Journal on Computing*, 14(1):134–147, February 1985.

[4] A. Flavell F. Ulgen and N. Akamatsu. Geometric shape recognition with fuzzy filtered input to a backpropagation neural network. *IEEE Trans. Inf. & Syst.*, E788-D(2):174–183, February 1995.

[5] H. Freeman and R. Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM*, 18(7):409–413, July 1975.

[6] Joaquim A. Jorge and Manuel J. Fonseca. A simple approach to recognise geometric shapes interactively. In *Proceedings of the Third Int. Workshop on Graphics Recognition (GREC'99)*, Jaipur, India, September 1999.

[7] Takayuki Dan Kimura, Ajay Apte, and Van Vo. Recognizing multistroke geometric shapes: An experimental evaluation. In *Proceedings of the ACM conference on User Interface and Software Technology (UIST'93)*, pages 121–128, Atlanta, GA, 1993. ACM Press.

[8] Joseph O'Rourke. *Computational geometry in C*. Cambridge University Press, 2nd edition, 1998.

[9] L. Schomaker. From handwriting analysis to pen-computer applications. *Electronics & Communication Engineering Journal*, June 1998.

[10] Charles C. Tappert, Ching Y. Suen, and Toru Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–807, August 1990.