

# A Simple Approach to Recognise Geometric Shapes Interactively

Joaquim A. Jorge

jorgej@acm.org

Manuel J. Fonseca

mjf@ist.utl.pt

Departamento de Engenharia Informática  
IST/UTL, Av. Rovisco Pais  
1049-001 Lisboa, PORTUGAL

## Abstract

*This paper presents a simple method to recognise multi-stroke sketches of geometric shapes. It uses temporal adjacency and global geometric properties of figures to recognise a simple vocabulary of geometric shapes including solid and dashed line styles, selection and delete gestures. The geometric features used (convex hull, smallest-area regular polygons, perimeter and area scalar ratios) are invariant with rotation and scale of figures. We have found the method very usable with acceptable recognition rates although the multi-stroke approach poses problems in choosing appropriate values for time-outs.*

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

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

## 1 Introduction

Recognition of hand-drawn geometric shapes has garnered new attention with the widespread adoption of Personal Digital Assistants (PDAs). 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 draftspeople or through computer peripherals. Recognising these shapes deals mainly with noise


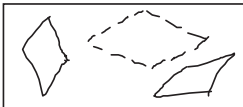






	Triangle		Diamond
	Rectangle		Ellipse
	Circle		Delete
	Line		Select

Figure 1: Geometric shapes and commands recognised.

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 recogniser 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 recognise dashed lines.

We want to emphasise that our main concern is to be able to recognise 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.

## 2 The Recognition Algorithm

The recognition algorithm is based on three main ideas. First, it uses entirely global geometric properties extracted from input shapes. Since we are solely interested in identifying geometric entities, the recogniser relies mainly on geometry information. Second, to enhance recognition performance, we use a decision tree to filter out unwanted shapes using distinctive criteria. Third, to overcome uncertainty and imprecision in shape sketches, we use fuzzy logic [1] to associate degrees of certainty to recognised shapes, thereby handling ambiguities naturally.

This algorithm recognises elementary geometric shapes, such as triangles, rectangles, diamonds, circles, ellipses and lines, and two gestures, delete and select, as depicted in Figure 1. Shapes are recognised independently of changes in rotation,

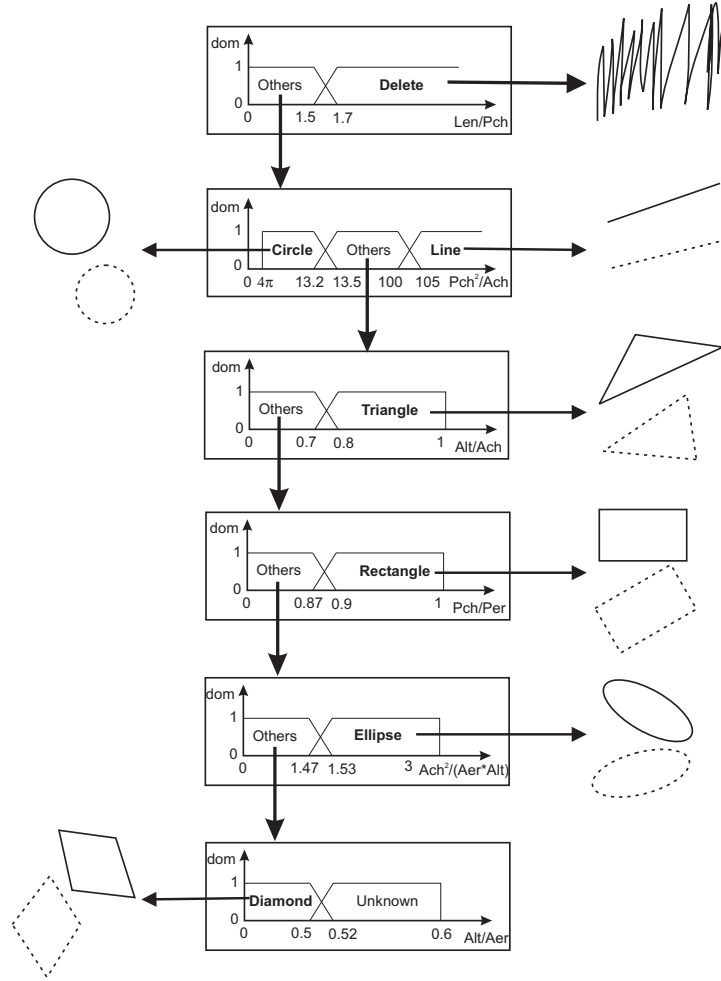
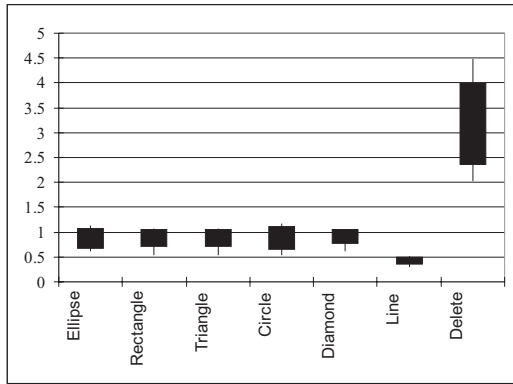


Figure 2: Decision tree and fuzzy sets.

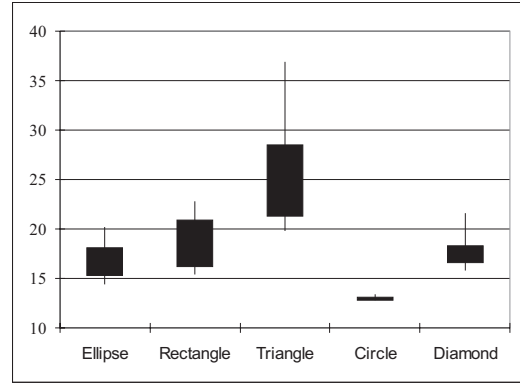
size or number of individual strokes. The recogniser works by looking up values of specific features in fuzzy sets at each decision tree node as illustrated by Figure 2. This traversal process yields a list of plausible shapes ordered by degree of certainty. This approach is largely based on a more restrictive method proposed by Kimura et. al.[4] which did not recognise non-rotated shapes, open/closed and dashed lines. Other authors have proposed more complex methods, involving neural networks[2] using a procedure that might prove more robust than ours, although such claims are not made explicit.

## 2.1 Feature Extraction

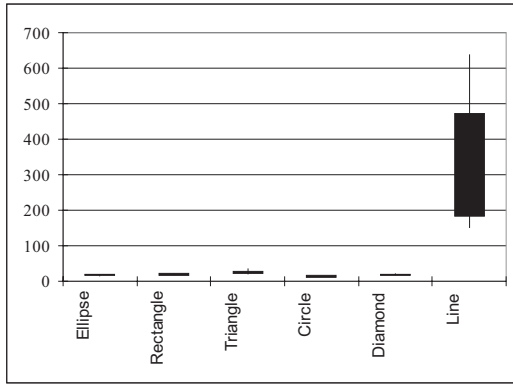
We start the recognition process by collecting data points using a digitising tablet, from the first pen-down event until a set timeout value after the last pen-up. Next, we compute the convex hull of the set of input points thus collected [5]. We use the convex hull to compute two special polygons. Using a simple three point algorithm we identify the largest-area triangle that fits inside the convex hull and the enclosing



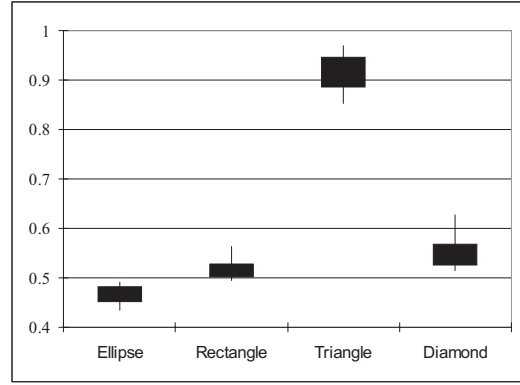
a - Len/Pch ratio.



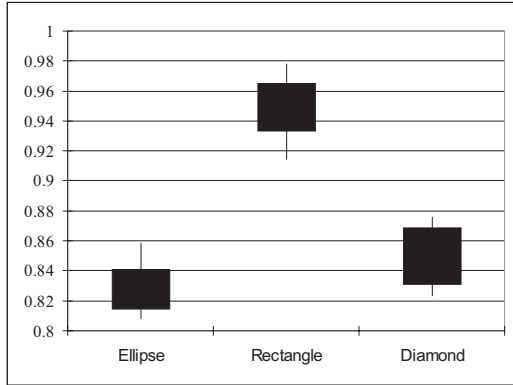
b - Thinness ratio for circles.



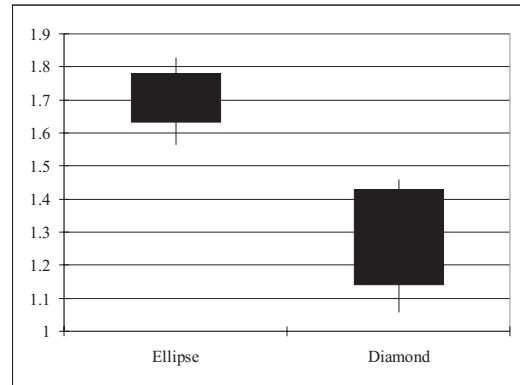
c - Thinness ratio for lines.



d - Alt/Ach ratio.



e - Pch/Per ratio.



f -  $Ach^2/(Aer*Alt)$  ratio.

Figure 3: Percentiles for area and perimeter ratios.

rectangle. Finally, we compute the area and perimeter of each polygon to estimate features and degrees of likelihood for each shape class.

Since this recogniser is intended for interactive use, we are interested in detecting gestures such as a set of zig-zag strokes drawn in succession to signify erasing objects underneath. Using our geometry approach, we detect this pattern by comparing the total length of strokes (**Len**) drawn to the perimeter of the convex hull (**Pch**). The ratio **Len/Pch** is close to unity for closed shapes, such as rectangles or circles. Large

values are associated to the **Delete** command (see Figure 3.a). Open shapes, such as polygonal lines and arcs, have  $\text{Len}/\text{Pch}$  values approaching one-half (typical of straight lines).

To distinguish circles and straight lines from other shapes we use the *Thinness ratio* ( $\mathbf{Pch}^2/\mathbf{Ach}$ ), where **Ach** is the area of the convex hull, and  $\mathbf{Pch}^2$  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  $4\pi$  (see Figure 3.b). On the other extreme, **Lines** have thinness values approaching infinity, as can be seen in Figure 3.c.

We identify triangles by comparing the area of the largest triangle (**Alt**) that fits inside the convex hull to that of the convex hull. The  $\mathbf{Alt}/\mathbf{Ach}$  ratio will have values near unity for **Triangles** and smaller values for other shapes (see Figure 3.d).

Similarly, let **Per** be the perimeter of the enclosing rectangle[3]. For rectangular shapes this perimeter will be very close to the convex hull's (cf. Figure 3.e), thus we use  $\mathbf{Pch}/\mathbf{Per}$  to distinguish **Rectangles** from ellipses and diamonds.

We have found ellipses and diamonds the hardest to distinguish. We use a more complex ratio involving the convex hull and the largest enclosed triangle and rectangle areas. The ratio  $\mathbf{Ach}^2/(\mathbf{Aer}*\mathbf{Alt})$  allows us to distinguish **Ellipses** which maximise it from diamonds which exhibit a smaller ratio (See Figure 3.f). Because diamonds take up about two times the area of a triangle, an area ratio ( $\mathbf{Alt}/\mathbf{Aer}$ ) under 50% means the object is a **Diamond**, otherwise the shape is not recognised. We should probably limit recognition of diamonds to constrained (*non-rotated*) shapes, for which the relationship above clearly holds.

To distinguish dashed from solid lines we use the ratio  $\mathbf{Pch}*\mathbf{NStrks}/\mathbf{Len}$  where **NStrks** is the number of strokes in the sketch. Because dashed shapes have a large number of strokes, with a small total length, this ratio exhibits large values for **Dashed** shapes and smaller values for closed shapes.

To choose the “best” fuzzy sets describing shape filters in the decision tree we used a “training set” developed by three subjects who drew each shape ten times using solid lines and five times using dashed lines. We used these values as depicted in Figures 3.a to 3.f to define the fuzzy sets used in the decision tree. Figure 2 shows the different fuzzy sets for each filter, the shapes that each identifies and the order of application of the different filters. Fuzzy sets were derived from the extreme percentiles from experimental values, trading-off rejections for false identification and increase in classification category, e.g. increasing the maximum thinness value for circles will “confuse” some ellipses with circles, or a decrease in the line minimum thinness value will cause thin shapes (such as rectangles) to be identified as lines.

## 2.2 Experimental Setup

In order to evaluate the recognition algorithm, we asked seven subjects to draw each shape fifteen times each, ten times using solid lines and five times using dashed lines. Each subject also drew the diagram presented in Figure 4.a, yielding a total of 934

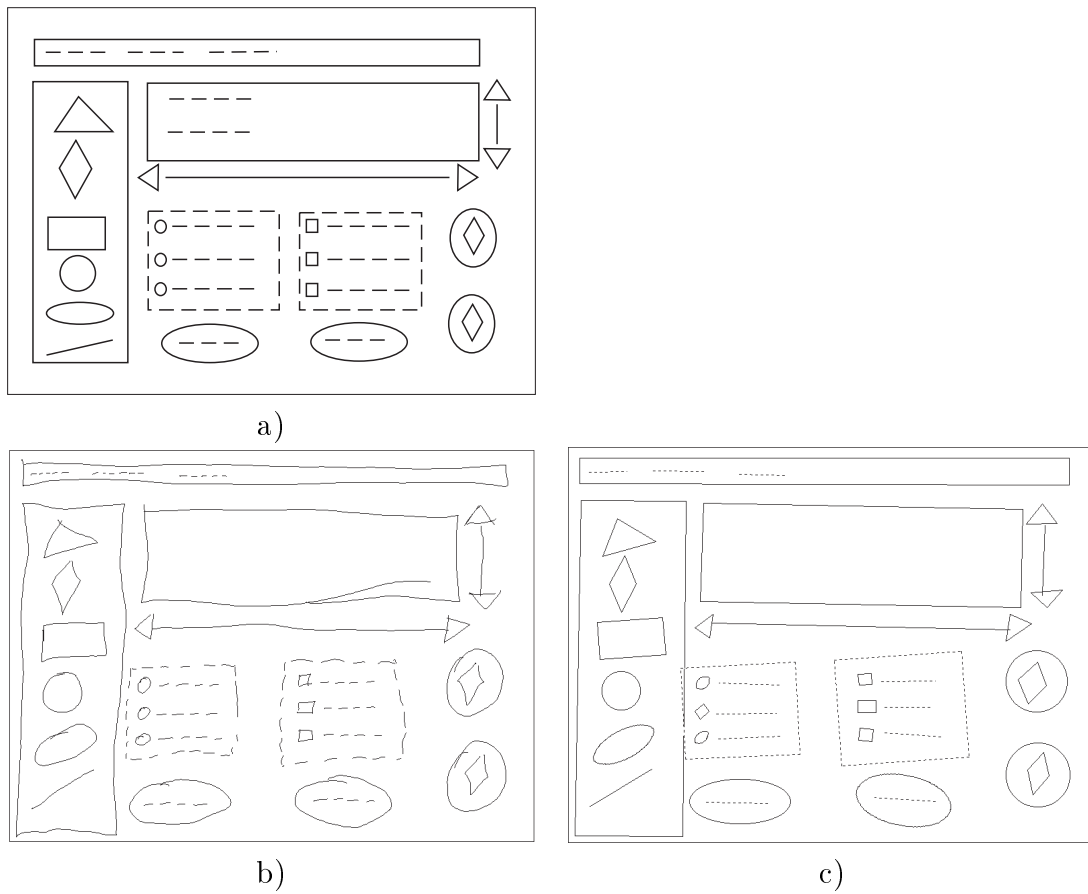


Figure 4: a) Diagram. b) Sketch. c) Recognised shapes.

shapes. Subjects were told that the experiment was meant to test recognition, so they didn't try to draw "unnatural" shapes. Figure 4.b shows a sample diagram drawn by one of the subjects and Figure 4.c shows the recognised shapes.

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

### 2.2.1 Results

The recogniser successfully identified 91% of the sketches drawn. The confusion matrix presented in Figure 5 shows the recognition rate for each shape category and the shapes wrongly recognised. A cursory analysis of the confusion matrix shows that circles are easily recognised as ellipses, which is both an acceptable and

		Recognised													
		Line S	Line D	Triangle S	Triangle D	Rectangle S	Rectangle D	Circle S	Circle D	Ellipse S	Ellipse D	Diamond S	Diamond D	Delete	Unknown
D r a w n	Line S	100%													
	Line D	3%	90%				5%								2%
	Triangle S			97%								2%			1%
	Triangle D				91%								9%		
	Rectangle S					96%				1%		3%			
	Rectangle D						96%			2%			2%		
	Circle S							73%		23%		3%			
	Circle D								83%		11%		6%		
	Ellipse S									85%		15%			
	Ellipse D										91%		9%		
	Diamond S					7%		1%		4%		88%			
	Diamond D						3%				9%		89%		
	Delete													100%	

Figure 5: Confusion matrix.

*intuitive* behaviour. If we consider circles and “fat” ellipses in the same shape class the recognition rate increases to 93%. Second, ellipses are confused with diamonds, and have a low recognition rate. Third, diamonds also have a low recognition rate and are confused with rectangles and ellipses.

If we consider successful classification when the “correct” shape is one of the top three shapes identified, the recognition rate increases from 91% to 94%. This recognition rate increases to 97% if we further consider circles and “fat” ellipses as the same shape class.

### 3 Discussion

We have described a simple and fast recogniser 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 recogniser in interactive settings shows that the observed recognition rates make it usable for drawing input. The time-outs required to recognise shapes regardless of the number of strokes using temporal adjacency have proved to be the most unnatural constraint so far.

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

## References

- [1] James C. Bezdek and Sankar K. Pal. *Fuzzy Models for Pattern Recognition*. IEEE Press, 1992.
- [2] A. Flavell F. Ulgen and N. Akamatsu. Geometric shape recognition with fuzzy filtered input to a bakpropagation neural network. *IEEE Trans. Inf. & Syst.*, E788-D(2):174–183, February 1995.
- [3] 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.
- [4] 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.
- [5] Joseph O'Rourke. *Computational geometry in C*. Cambridge University Press, 2nd edition, 1998.
- [6] L. Schomaker. From handwriting analysis to pen-computer applications. *Electronics & Communication Engineering Journal*, June 1998.
- [7] 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.