Experimental Evaluation of an on-line Scribble Recognizer

Manuel J. Fonseca [1] Joaquim A. Jorge

Departamento de Engenharia Informática, IST/UTL Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal

Abstract

We present a fast, simple and compact approach to recognize scribbles (multi-stroke geometric shapes) drawn with a stylus on a digitizing tablet. Regardless of size, rotation and number of strokes, our method identifies the most common shapes used in drawings, allowing for dashed, continuous or overlapping strokes. Our method combines temporal adjacency, Fuzzy Logic and geometric features to classify scribbles with measured recognition rates over 97%.

Key words: On-line Shape Recognition, Tools and Techniques, Fuzzy Logic, Multi-stroke Shapes, Experimental Evaluation

1 Introduction

User Interfaces are developing away from the classical WIMP (Windows, Icons, Mouse and Pointing) paradigm. Among the new generation of intelligent systems there is a broad class based on new modalities such as sketching, hand drawing and stylus input gestures, which we call *Calligraphic Interfaces*. These seem natural, because people use diagrams to think, to represent complex models and ideas, to reason about and discuss designs in domains as diverse as architecture, engineering and music. Despite the "G" in the current generation of Graphical User Interfaces, these instances are characterized by discrete interaction modalities such as selecting items from menus, pointing to things on the screen and pressing buttons. The drawing paradigm, so powerful and so useful for humans goes largely underutilized.

[1] Corresponding author. Fax: +351 21 3145843; E-mail: mjf@inesc.pt

Our aim is to develop simple components to help people use computers in more engaging and "natural" modes. To this end, we have developed a simple recognizer capable of identifying the most common constructs and gestures used in drawings. In this paper we review the state of the art in on-line graphics recognition, describe the geometric features used and examine the experimental results from evaluating our approach. While the method uses very simple concepts and a reduced set of geometric features, it is surprisingly robust and extensible, making it usable in a wide variety of practical environments.

2 Related Work

The idea of calligraphic interfaces is not new, even if we discount that sketches and planar diagrams as a way of communication precede the invention of writing by more than 30 centuries (Harley and Woodward, 1987). Sutherland (1963) presented Sketchpad, the first interactive system that used one light pen to draw diagrams directly over a screen surface. The main limitation of this system resided in the recognition capabilities, limited resources and high cost of the computer used.

Nevertheless, due in part to ergonomic problems with the light pen and the invention of mouse in 1964, current graphical interfaces relegated pen-based interactions to specific CAD applications. Things changed with the appearance of the first pen computers in 1991, even though pens were used mainly to input text through handwriting recognition.

The Newton system (Kounalakis and Menuez, 1993), one of the first handheld pen based computers, incorporates handwriting, shape and gesture recognizers. While the shape recognizer only handles uni-stroke, axis-aligned shapes, the gesture recognizer is more suitable for text editing than for drawing schematics, which are the main scope of our work.

Even though handwriting recognition is not the main target of our work, we will shortly analyze the Graffiti (Blinkenstrofer, 1995) system. This system recognizes characters drawn with a single stroke. While this makes the recognition process easier, on the other hand it forces users to learn a new vocabulary. Graffiti, as is typical with most handwriting recognition systems, uses local features, such as drawing speed, while our method uses global geometric information to characterize drawings. As a result, our recognition process uses simpler features and analyzes complex shapes as a whole, regardless of the way individual strokes were drawn. Further, our vocabulary is considerably smaller than Graffiti's, thus requiring a much smaller number of features.

Our work is based on a first approach to recognize schematic drawings devel-

oped at the University of Washington (Apte et al., 1993), which recognized a small number of non rotated shapes and did not distinguish open/closed, dashed or bold shapes. Tappert et al. (1990), provide an excellent survey of the work developed in the area of non-interactive graphics recognition for the introduction of data and diagrams in vectorial CAD systems. Zhao (1993) describes an interactive editor based on the StateCharts (Harel, 1987) formalism. Although the editor uses a mouse and direct manipulation, many of the ideas described by Zhao express an approach based on diagram recognition guided by syntax.

Gross (1996) describes a system fundamentally based on sketches that are partially interpreted for use in architecture drawings, using a recognizer substantially simpler and less robust than ours. Although this recognizer is trainable, the number of identified shapes is apparently smaller than ours and they can only be rotated by multiples of 90^o because of the zoning approach used. Landay (1996), uses a recognizer developed by Rubine (1991) to sketch graphical arrangements of bidimensional documents. Although Rubine's recognizer is trainable, it does not achieve a high recognition rate to ensure a consistent use. Despite the simplicity of our recognizer and the fact it is not trainable, it allows the identification of a larger number of geometric shapes robustly, making it a complementary approach to Rubine's work.

Other authors have proposed more complex methods, involving neural networks (Ulgen et al., 1995), to identify a small number of geometric shapes (rectangles, squares, circles, ellipses and triangles). These shapes are recognized independently of size and rotation, but they can not be drawn using dashed or bold lines. Even though the authors claim good performance for their method, the paper does not present clear measurements to back their claims.

3 The Recognition Algorithm

The recognition method is based on three main ideas. First, we use entirely global geometric properties extracted from input shapes, because we are interested in identifying geometric entities. Second, to enhance recognition performance, we use a set of filters either to identify shapes or to remove unwanted shapes using distinctive criteria. Third, to overcome uncertainty and imprecision in shape sketches, we use fuzzy logic (Bezdek and Pal, 1992) 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 ges-

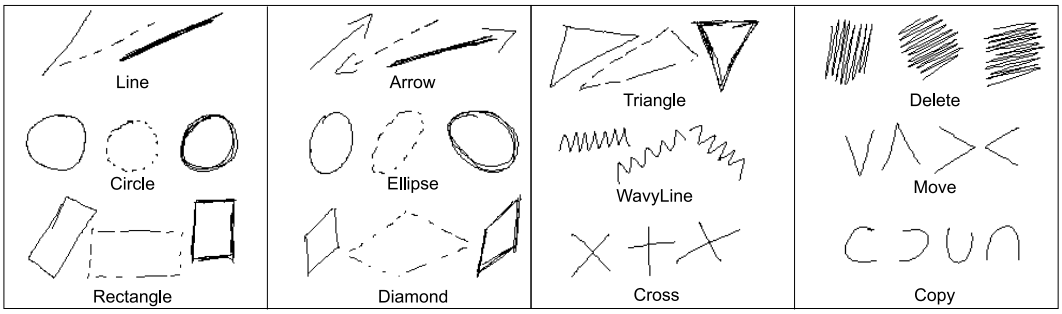

Fig. 1. Multi-stroke and uni-stroke shapes.

ture commands, `Delete`, `Cross`, `WavyLine`, `Move` and `Copy`, as depicted in Fig. 1. Shapes are recognized independently of rotation, size or number of strokes.

The set of shapes selected and presented in Fig. 1 are the basic elements to allow the construction of technical diagrams such as electric or logic circuits, flowcharts or architectural sketches. These diagrams also require distinguishing between solid, dashed and bold depictions of shapes in the same family. Typically, architects will use multiple overlapping strokes to embolden lines in sketches, a mechanism 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.

The algorithm works by collecting strokes from a digitizing tablet. A *stroke* is the set of points from pen-down to pen-up. We collect strokes into *scribbles* until a set *timeout* value is reached. Recognized scribbles are classified as *shapes*. A shape associates a scribble with a class (e.g. `Triangle`) and a set of geometric attributes, such as start-, end-points and bounding box. The recognizer works by looking up values of specific features in fuzzy sets associated to each shape. This process may yield a list of plausible shapes ordered by degree of certainty. If the recognizer can not associate any shape to the scribble, it will return the `Unknown` shape, toghether with some basic geometric attributes, such as linestyle and "openness". The original scribble is always available as an attribute of the recognized shape, for use in drawing applications.

The present is an evolution of previous work (Jorge and Fonseca, 1999), which did recognize less shapes and used a decision tree to prune out incorrect classifications. We discovered that using classification rules alone, provides more flexibility and extensibility without sacrificing robustness.

The next subsections describe how we select geometric features, derive fuzzy sets from experimental data and obtain the final classification results. We also discuss re-segmentation as a way to improve recognition of specific shapes

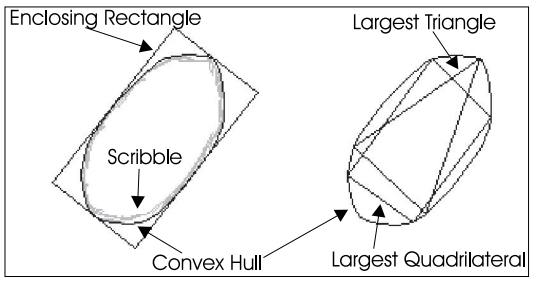

Fig. 2. Polygons used to estimate features.

such as `Arrows` and `Crosses`. Finally, we discuss how our approach handles ambiguous shapes in a natural manner.

3.1 Geometric Features

After collecting each scribble from the digitizing tablet, we compute the convex hull (ch) of its points, using Graham's scan (O'Rourke, 1998). We then use this convex hull to compute three special polygons. The first two are the largest area triangle (lt) and quadrilateral (lq) inscribed in the convex hull (Boyce and Dobkin, 1985). The third is the smallest area enclosing rectangle (er) (Freeman and Shapira, 1975) (see Fig. 2). Finally we compute the area and perimeter for each special polygon, to estimate features and degrees of membership for each shape class.

To select 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 of a feature 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 prevent unwanted classifications, yielding more complex rules as needed. The main features we use are ratios between perimeters and areas of the special polygons described above.

Fig. 3 shows the percentile graphics for features P_{ch}^2/A_{ch} and H_{er}/W_{er}. The first feature, *Thinness* ratio, where A_{ch} is the area of the convex hull and P_{ch}^2 is its perimeter squared, is used to distinguish `Circles` from other shapes. 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. 3a). In this

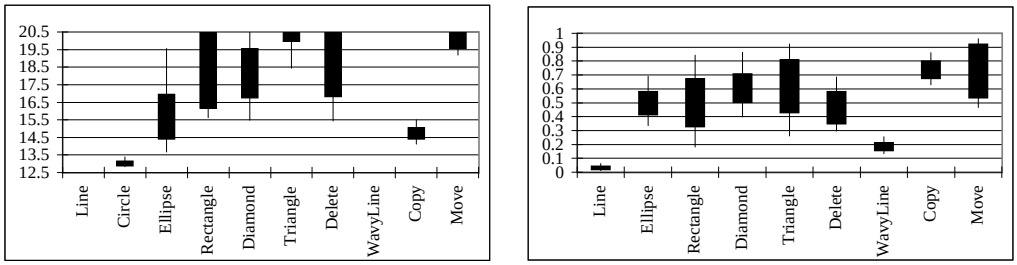

Fig. 3. a) Percentiles for the P_{ch}^2/A_{ch} ratio. b) Percentiles for the H_{er}/W_{er} ratio.

figure the thinness values for `Lines` 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 the aspect ratio, which compares the height of the 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. 3b). The rest of the features were selected using a similar method of analyzing all percentile graphics, to identify unique traits for each shape.

3.2 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 the polygons described above.

Each shape is defined by several fuzzy sets, some of which identify the shape while others serve to avoid "wrong" results. We do not use the same number of features for all shapes. Some shapes require no more then one or two features. In general we tend to avoid using "too many" fuzzy sets to characterize a shape. This 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 alone.

A fuzzy set is defined by four values, as depicted in Fig. 4. After selecting the features for each shape, we compute these four values based on the percentiles. Values `b` and `c` correspond to the 10% and 90% percentiles respectively, and `a` and `d` to the "minimum" and "maximum" after we have identified outliers in each distribution. Removing these outliers minimizes confusion between different shape families, which rises from overlap in distributions. Thus, there is a tradeoff in designing fuzzy sets from statistical data. If `a` and `d` are set

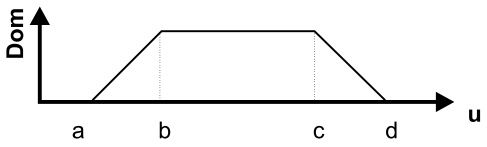

Fig. 4. Definition of a fuzzy set.

very wide apart the recognition rate increases, as well as the number of false positives. "Narrower" values decrease the number of false classifications at the cost of a much lower recognition rate. Abe and Lan (1996) discuss an automated procedure for collecting this information using activation and inhibition hyper-boxes. We chose to generate rules manually, since their approach is not sufficiently flexible for our purposes.

3.3 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`ing together degrees of membership for the relevant fuzzy sets.

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 A Line
```

```
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 A Diamond
```

The rule on the left is the simplest rule used in our recognizer. It ascertains that "very thin" ($H_{er}/W_{er} \simeq 0$) scribbles should be classified as `Lines`. A more complicated rule recognizes `Diamonds`, where `AND` denotes the conjunction of fuzzy predicates $\mu_x(f \texttt{AND}\ g) = \min(\mu_x(f), \mu_x(g))$ and `NOT` is defined by $\mu_x(\texttt{NOT}\ f) = 1 - \mu_x(f)$.

The algorithm distinguishes between `Solid`, `Dashed` and `Bold` styles after identifying "basic" shapes. This enables us to treat linestyle as an attribute orthogonal to shape, making the design of our recognizer more modular and easier to add different shapes in the future.

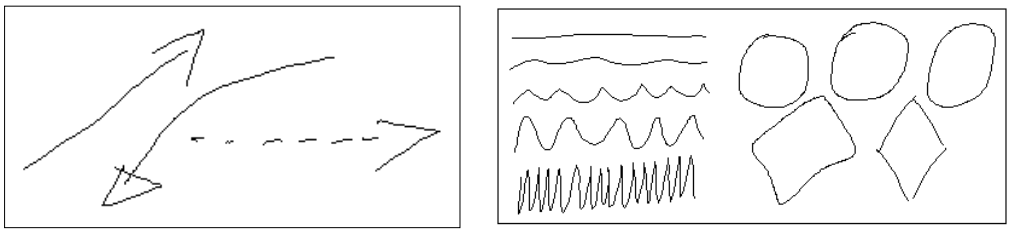

Fig. 5. a) Different types of `Arrows`. b) Ambiguity cases among shapes.

3.4 Re-segmentation

An approach based entirely on global geometric properties has some limitations. Even though `Arrows` or `Crosses` cannot be recognized using this approach, it would be useful if our recognizer identified them, since they are commonly used in most diagram notations. In order to recognize these shapes we must look for new properties that characterize them. Among those, we can consider the small number of strokes, or the existence of a stroke that uniquely identifies the shape, or a distinct spatial relation between strokes. For example `Arrows` are built of a variable set of strokes terminated by a last stroke which is either a `Triangle` or a `Move` shape, as shown in Fig. 5a. Finally, a `Cross` consists of two intersecting strokes (that must be `Lines`).

The next two rules show how we classify these shapes. The first rule identifies `Arrows` while the second defines `Crosses`.

```
IF NumStrokes >= 2 AND
   (LastStrk IS LIKE Triangle OR
   LastStrk IS LIKE Move)
THEN
     Shape IS A Arrow
```

```
IF NumStrokes == 2 AND
   FirstStrk IS LIKE Line AND
   SecondStrk IS LIKE Line AND
   FirstStrk INTERSECT SecondStrk
THEN
     Shape IS A Cross
```

We perform the analysis of these new properties by re-segmenting the original scribble and by applying the recognition process to some specific strokes, e.g. the last stroke in the `Arrow`. Re-segmentation allows recognizing new gestures that could not be identified using just geometric properties.

3.5 Ambiguity

Considering the shapes identified by the recognizer, we present four special cases which can yield ambiguous results. Ambiguity exists between `Lines` and `WavyLines`, `WavyLines` and `Deletes`, `Circles` and `Ellipses`, `Diamonds` and `Rectangles`. These cases are presented in Fig. 5b.

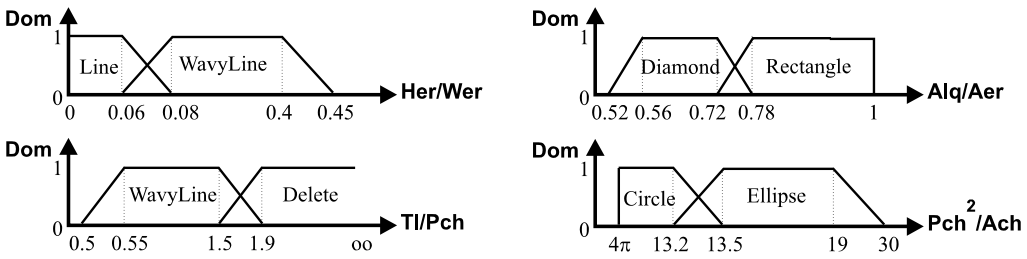

Fig. 6. Fuzzy sets representing the ambiguity cases supported by the recognizer.

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

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

The ambiguities between shapes are modeled naturally using fuzzy logic to associate degrees of certainty to recognized shapes. Fig. 6 illustrates corresponding fuzzy sets for the ambiguous cases shown in Fig. 5b.

4 An example: Calligraphic Editor

The Calligraphic Editor presented in this section uses the recognizer described above and allows the creation of "beautified" diagrams from "imperfect" sketches. With this editor, users can create geometric shapes and invoke commands using hand drawn sketches. Shapes can be drawn, copied, moved and resized, using just the pen. We can delete an object or a set of objects by drawing the gesture command `Delete`, as illustrated in Fig. 7a.

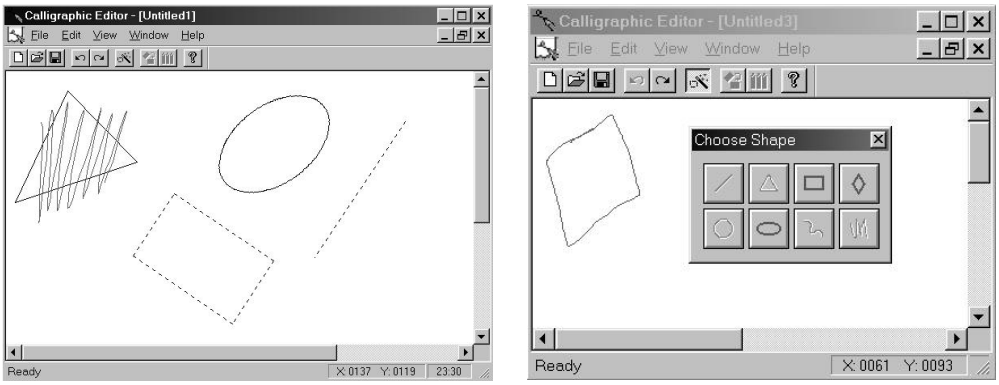

Fig. 7. a) Deleting a shape. b) Menu showing ambiguous shape recognition.

This editor uses a simple way to deal with the ambiguity afforded by the recognizer. Whenever the recognizer returns more than one shape, as a result, the editor displays a menu with all the possibilities, as shown in Fig. 7b, allowing the user to choose the "correct" alternative. This approach makes the creation of diagrams easier, eliminating the need for redrawing the same sketch, to get the desired result.

We are planning to try other solutions, such as using different colors to communicate multiple alternative shapes or to denote the degree of certainty of the result. When the recognition process does not yield a unique result we should offer an interaction technique to allow circulating among all returned shapes. Mankoff and Abowd (1999) describe some of these mechanisms to deal with ambiguity and errors present in handwriting recognition systems.

5 Experimental Results

To evaluate the recognition algorithm, we asked nine subjects to draw each multi-stroke shape 40 times, using solid, dashed and bold lines, 30 times each uni-stroke shape and a simple Entity/Relationship diagram with 22 shapes. All these drawings yield a total of 4068 shapes. 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. Two 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 with rotation or size and about the set timeout value. Novice subjects had a short practice session in order to become acquainted to the stylus/tablet combination. During the drawing session the recognizer was turned off in order not to interfere with data collection and to avoid any kind of adaptation from the user.

The recognizer successfully identified 95.8% of the scribbles drawn considering just the first shape identified. It is fast: each scribble requires, on average, less than 50 ms (using a Pentium II @ 233 MHz) to be recognized, from feature vector computation to final classification.

A cursory analysis of the confusion matrix, shown in Fig. 8, reveals that `Diamonds` are often confused with `Rectangles`, and have the lowest recognition rate. `Arrows` are other shape which exhibits low recognition rate. The former is due to the ambiguity between `Rectangles` and `Diamonds` that favors

		Recognized												
	Shapes	Line	Arrow	Triangle	Rectangle	Diamond	Circle	Ellipse	Delete	WaviLine	Copy	Move	Cross	Unknown
Drawn	Line	**97.7**			0.5				1.4	0.2				0.2
	Arrow	0.5	**90.2**	0.3						0.3				8.7
	Triangle	0.3	0.8	**97.8**						0.6				0.6
	Rectangle	0.2	0.5		**96.9**	1.4		0.2						0.7
	Diamond		0.8		8.4	**87.8**		0.3	0.3	0.5				1.9
	Circle				0.3		**97.2**	2.5						
	Ellipse						1.1	**97.6**	0.5					0.8
	Delete					0.4			**98.9**					0.7
	WavyLine	2.0							2.7	**94.6**				0.7
	Copy										**99.6**	0.4		
	Move			3.0								**94.8**		2.2
	Cross												**97.0**	3.0

Fig. 8. Confusion matrix (values in percentages).

`Rectangles` and the latter is due to incorrect drawing (single-stroke) of arrows by users. We can also identify other cases of confusion between shapes, such as `Circles` with `Ellipses`, `Moves` with `Triangles` and finally `WavyLines` with `Lines` and with `Deletes`. In fact, the confusion between these shapes is both an acceptable and *intuitive* behavior.

Since ambiguity is one of the main characteristics of our recognizer, we prefer to consider the top three shapes identified instead of just the most likely one. Using this when we take this route, the recognition rate increases to 97%, showing a good improvement relatively to our previous approach (Jorge and Fonseca, 1999) that had a recognition rate of 94% using less shapes and a more complex method.

6 Conclusions

We have described a simple and fast recognizer for elementary geometric shapes. An in-depth description of our approach can be found in (Fonseca and Jorge, 2000), while the source code is publicly available under the GNU GPL at `http://immi.inesc.pt/~mjf/cali`. 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. We improved on previous work (Jorge and Fonseca, 1999) by increasing recognition rates using fuzzy rules instead of decision trees and introducing resegmentation to identify higher-level patterns such as arrows. 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 (Nauck and Kruse, 1997)

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.

References

Abe, S., Lan, M.-S., 1996. Efficient Methods for Fuzzy Rule Extraction From Numerical Data. In: C. H. Chen (Ed.), Fuzzy Logic And Neural Networks Handbook. IEEE Press, pp. 7.1–7.33.

Apte, A., Vo, V., Kimura, T. D., 1993. Recognizing Multistroke Geometric Shapes: An Experimental Evaluation. In: Proceedings of UIST'93. Atlanta, GA.

Bezdek, J. C., Pal, S. K., 1992. Fuzzy Models for Pattern Recognition. IEEE Press.

Blinkenstrofer, C. H., 1995. Graffiti. Pen Computing , 30–31.

Boyce, J. E., Dobkin, D. P., Feb. 1985. Finding Extremal Polygons. SIAM Journal on Computing 14 (1), 134–147.

Fonseca, M. J., Jorge, J. A., Jul. 2000. CALI: A Software Library for Calligraphic Interfaces. Tech. rep., DEI/IST/Technical University of Lisbon.

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

Gross, M. D., 1996. The Electronic Cocktail Napkin - A computational environment for working with design diagrams. Design Studies 17 (1), 53–69.

Harel, D., 1987. StateCharts: A visual formalism for complex systems. Science of Computer Programming 8, 231–274.

Harley, J. B., Woodward, D. (Eds.), 1987. The History of Cartography. Vol. 1. University of Chicago Press, Chicago, IL.

Jorge, J. A., Fonseca, M. J., Sep. 1999. A Simple Approach to Recognise Geometric Shapes Interactively. In: Proceedings of the Third Int. Workshop on Graphics Recognition (GREC'99). Jaipur, India.

Kounalakis, M., Menuez, D., 1993. Defying Gravity: The Making of Newton. Beyond Words Pub Co.

Landay, J. A., Dec. 1996. Interactive sketching for the early stages of user interface design. Ph.D. thesis, Carnegie Mellon University, Computer Science, Pittsburgh - USA.

Mankoff, J., Abowd, G. D., Jun. 1999. Error Correction Techniques for Handwriting, Speech, and other ambiguous or error prone systems. Tech. rep., GVU TechReport GIT-GVU-99-18.

Nauck, D., Kruse, R., 1997. A neuro-fuzzy method to learn fuzzy classification rules from data. Fuzzy Sets and Systems (89), 277–288.

O'Rourke, J., 1998. Computational geometry in C, 2nd Edition. Cambridge University Press.

Rubine, D. H., Dec. 1991. The automatic recognition of gestures. Ph.D. thesis, Carnegie Mellon University, Computer Science, Pittsburgh - USA.

Sutherland, I. E., 1963. Sketchpad: A Man-Machine Graphical Communication System. In: Spring Joint Computer Conference. AFIPS Press.

Tappert, C. C., Suen, C. Y., Wakahara, T., August 1990. The state of the art in on-line handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 12 (8), 787–807.

Ulgen, F., Flavell, A., Akamatsu, N., February 1995. Geometric shape recognition with fuzzy filtered input to a backpropagation neural network. IEICE Trans. Inf. & Syst. E78-D (2), 174–183.

Zhao, R., 1993. Incremental Recognition in Gesture-Based and Syntax-Directed Diagram Editors. In: Proceedings of INTERCHI'93. Amsterdam.