Smart Sketchpad—An On-line Graphics Recognition System

Liu Wenyin[1], Wenjie Qian[1], Rong Xiao[2], Xiangyu Jin[2]
[1]*Microsoft Research China, 49 Zhichun Road, Beijing 100080, PRChina*
wyliu@microsoft.com
[2]*State Key Lab for Novel Software Tech., Nanjing University, Nanjing 210093, PRChina*
{cloud, jxy}@graphics.nju.edu.cn

Abstract

An on-line graphics recognition system is presented, which provides users a natural, convenient, and efficient way to input rigid and regular shapes or graphic objects (e.g., triangles, rectangles, ellipses, straight line, arrowheads, etc.) by quickly drawing their sketchy shapes in single or multiple strokes. An input sketchy (hand-drawn) shape is immediately converted into the user-intended rigid shape based on the shape similarity and the time constraint of the sketchy line. Three different (rule-based, SVM-based, and ANN-based) approaches have been applied and compared in the system. Experiments and evaluation are also presented, which show good performance of the system.

Keywords: Graphics recognition, sketch recognition, shape recognition, pen-based user interface, support vector machines (SVMs), artificial neural network (ANN)

1. Introduction

Graphics is an important means of expression. Many computer systems have to handle graphics in many functions, including input, processing, and rendering. In such systems, including Microsoft VISIO, Office, PhotoDraw, and many CAD systems, graphics input is an inevitable requirement. The most popular way of drawing graphic objects is using mouse/keyboard with lots of toolbar buttons or menu items for selection. In this way, users are allowed to directly draw/input regular shapes or graphic objects, such as those in many diagrams. However, this is not the natural and convenient way for humans to draw graphics. In order to adapt such systems to users, pen/tablet devices are invented as an important extension of mouse/keyboard for input. However, they are mainly used for handwriting character input or for replacement of the mouse during directly drawing regular shape graphic objects. It is especially non-realistic to draw graphics in such way on small screen devices, such as PDAs. The reason is that, although pens are used, the toolbar buttons and menu items make the drawing area on the screen even smaller. The most convenient and natural way for human beings to draw graphics should be to use a pen to draw sketches, just like drawing on a real sheet of paper. Moreover, it is even better to recognize and convert the sketchy curves drawn by the user to their rigid and regular shapes immediately. This is because, with the on-line graphics recognition as an immediate and useful feedback, the user can realize errors or inappropriateness earlier and therefore draw the diagrams more perfectly. In this paper, we refer to the approach and process of immediately and automatically converting the input sketchy curve to a rigid and regular geometry shape as on-line graphics recognition.

As a main application of pen/tablet input, on-line handwriting recognition (Yaeger 1996) is very common to many users and its prevalence is increasing. However, very few research works have been done on on-line graphics (or hand-drawing, or sketch) recognition. Zeleznik et al. (1996) have invented an interface to input 3D sketchy shapes by recognizing the defined patterns of some input 2D graphics that correspond to certain sketchy solid shapes. Some sketchy input interfaces, such as the one developed by Gross and Do (1996), are mainly for conceptual and creative layout designs. Ajay et al. (1993) have proposed an efficient and very easy graphics recognition algorithm based on filters. Their approach can handle six types of geometric shapes, including triangles, rectangles, ellipses, circles, diamonds, and lines. However, these filters are sensitive to orientation of the geometric objects. Only circles and lines are orientation independent. All the other shapes should be drawn in parallel with X-axis and Y-axis. Although they have achieved very high recognition precision (98%) in their experiments, their precondition is somewhat too strict. Fonseca and Jorge and their group (Fonseca and Jorge 2000; Albuquerque et al. 2000) have extended Ajay et al.'s work by adding some new filters. Fuzzy logic is also employed in their graphics recognition approach such that their recognition approach is orientation independent. Based on global area and perimeter calculation, these filters can hardly distinguish ambiguous shapes such as pentagon and hexagon. The sketch recognition work reported by Arvo and Novins (2000) is mainly for 2D on-line graphics recognition. Their approach continuously morphs the sketchy curve to the guessed shape while the user is drawing the curve. However, their

main purpose focuses on the user studies of such sketch recognition system. Moreover, their recognition approach only handles two simplest classes of shapes (circles and rectangles) drawn in single strokes. This is not adequate for a real software tool that can be used for inputting most classes of diagrams. Anyway, their preliminary conclusion shows that such interface is particularly effective for rapidly constructing diagrams consisting of simple shapes. Hence, in order to provide the capability to input more complex diagrams, it is necessary to extend the sketchy graphics recognition approach to handle more classes of shapes, including lines (either straight or free-formed) and arrowheads, triangles, and other complex shapes, in more general cases.

In this paper, we proposed an on-line graphics recognition algorithm and implement it in our Smart Sketchpad system. It integrates three different (rule-based, SVM-based, ANN-based) approaches and can handle more classes of shapes that may consist of multiple strokes. The recognizable graphic objects include free curves, straight lines, ellipses (circles), triangles (with different types: scalene triangles, equilateral triangles, isosceles triangles, right-angled triangles, acute/obtuse triangle), rectangles (squares, and general quadrilaterals, trapezoids, parallelograms, rhombuses), and arrowheads. Other shapes will be added in the future. Experiments and evaluation are also conducted and presented in this paper.

2. The On-Line Graphics Recognition Algorithm

Considering some real user scenarios of drawing graphic objects as if using the pen/paper interface, we construct the following on-line graphics recognition algorithm.

1. Input: The input is a stroke drawn by a user, which is a trajectory of the pen movement on a tablet between the time when the pen-tip begins to touch the tablet and the time when the pen-tip is lifted up from the tablet. The stroke is represented in a chain of points. Go to Step 2.
2. Pre-processing: In this process, the stroke is first polygonalized to a polyline (an open polygon), the two endpoints of which are then refined and polished to remove noises. Go to Step 3 (open/closed test).
3. Open/closed polyline test: A polyline is considered nearly closed if its two endpoints are close enough (determined by an adaptive threshold, e.g., the smaller of 20 pixels and 1/8 of its length). If the polyline is closed, go to Step 6 (shape recognition). Otherwise (not closed), go to Step 4 (linkable test).
4. Linkable test: We assume that two strokes can be linked up if the distance between their nearest endpoints is less than a threshold, which is set to 20 pixels in this paper. If the current stroke can be linked with one or more existing strokes, link them up to form a new polyline and go to Step 3 (open/closed test) with the new polyline. Otherwise, go to Step 5 (open graphic object finalization)
5. Open graphic object finalization: The current polyline is finalized and classified as one of the three types of open graphic objects: a straight line (if only two endpoints remain), a multiple-edge polyline (if more than one intermediate points between the two endpoints and the edges are longer than a certain threshold), or a free-formed curve (which is rendered with its original stroke if it is not of the other two types). Go to Step 8 (compound graphic object recognition) with the finalized open graphic object.
6. Closed shape recognition: The closed polyline is sent to the shape recognition procedure to determine its shape type as one of the three known types: triangle, rectangle, and ellipse. Go to Step 7.
7. Parameter estimation: The parameters (e.g., polygon vertices or ellipse foci and axes) of the recognized shape are estimated approximately such that the recognized graphic object fits the sketchy input shape to the best extent. Go to Step 8.
8. Compound graphic object recognition: Check if the recognized single graphic object (either a closed shape or an open polyline) can be combined with existing graphic objects (either closed shapes or single strokes) to form a compound graphic object. If a new compound graphic object is recognized, go to Step 8. Otherwise, go to Step9.
9. Stop.

In the above algorithm, there are several procedures that will be elaborated in detail in the following sections. The pre-processing process is presented in Section 3. The closed shape recognition process is presented in Section 4. The parameter estimation (post-processing) process is presented in Section 5. The compound graphic object recognition process is presented in Section 6.

3. Pre-processing

The pre-processing process includes two steps: stroke polygonalization (or polygonal approximation) and endpoint refinement.

3.1. Polygonal Approximation

After input, the sketchy line of the stroke is represented by a chain of points. Due to nonproficiency or un-professionalism, the sketchy line is usually very curvy and free-formed. For example, without using a ruler, a straight line drawn by a drafter is not so straight if measured strictly no matter how much attention the drafter is paying to the drawing operation. The case of sketching a circle is similar. In many cases, some intermediate points on the sketchy line are redundant because they lie (approximately) on the straight-line

segment formed by connecting their neighbors. These points are called non-critical points and should be removed from the chain so that the sketchy line can be approximately represented by a polyline (an open polygon) with the fewest critical vertices. The polyline should maintain the original shape to a certain extent that is controlled by a parameter ε. This procedure is called polygonalization or polygonal approximation, which removes as many non-critical points as possible from the point chain, such that the maximal distance of any non-critical point between every pair of consecutive critical points to the chord joining these this pair of points is less than ε. We apply the algorithm developed by Sklansky and Gonzalez (1980) to implement the polygonal approximation process. In order to preserve well the original shape of the sketchy line and to remove the non-critical points in a well-proportioned manner, we apply a two-pass polygonal approximation process. In the first pass, we use ε=1.0 pixel and in the second one we use ε=5.0 pixels, as shown in Figure 1(a). Note that if we simply apply one pass polygonal approximation process with ε=5.0 pixels, the lengths of the remaining edges of the polyline are usually not uniform and the shape is therefore severely distorted due to cumulative error tolerances.

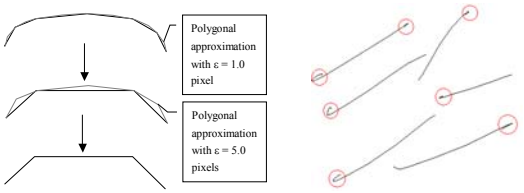


Figure 1. (a) Illustration of the two-pass polygonal approximation process. The gray lines are before each pass and the black lines are after each pass. (b) Some examples of imperfect endpoints of strokes.

3.2. Endpoint Refinement

Due to the shaky operations caused when the pen-tip touches the tablet and when it is lifted up, there are often some hooklet-like segments at the endpoints of the sketchy lines. Such cases happen especially when the sketchy line is drawn quickly (i.e., completed in a very short time). See Figure 1(b) for some examples of such imperfect endpoints of strokes. These segments are just noises and usually remain after polygonal approximation. They should be removed in order to correctly recognize the sketchy line/curve as a regular shape of the graphic object. Therefore, we refine the endpoints of the obtained polyline before the regular shape recognition process starts. Currently, endpoint refinement approach is simply rule-based. Empirically, if the first edge starting from the current endpoint of the polyline forms an angle of less than 90 degrees with its neighbor edge and its length is less than half of its neighbor edge, this edge is removed and the next point becomes the new endpoint of the polyline.

4. Closed Shape Recognition

After we get the noise-free and polygonalized stroke from the previous section, we analyze its shape similarity to known graphic shapes. Currently we have three different recognition approaches: rule-base, SVM-based, and ANN-based, which are presented in the following subsections.

4.1. Rule-Based Sketchy Graphics Recognition

The most intuitive method to determine the shape type is to examine the remaining vertices and edges the polyline. Long edges are most likely part of the real edges of the recognized triangle or rectangle. Another factor that affects the shape recognition is the completion time of the sketchy stroke. There are some intuitive rules to determine the shape types. For instances, if the sketchy line is finished quickly (e.g., less than half second) and there are several vertices (e.g., the number of vertices is greater than 8) in the approximate polyline, it is more likely an ellipse than a rectangle or a triangle. It is obviously a triangle if 4 vertices are left (note that there are two endpoints of the polyline since the polygon is open). However, if more than 4 vertices are left, it is still possible to be a triangle.

For those cases in which the shape cannot be determined by the intuitive rules, we use the Hypothesis-and-Test strategy to determine the shape type. We first hypothesize that it is a triangle, a rectangle, or an ellipse, the parameters of which is then estimated using the approaches described in Section 5 to finalize the recognized graphic object. We calculate the difference between each hypothesized graphic object and the original sketchy stroke. The difference is the average of the minimal distance of each vertex of the original stroke to the hypothesized graphic object. The shape with the smallest average distance is selected as the shape recognition result.

4.2. The SVM-Based Shape Recognition Approach

Support Vector Machine (SVM) is a new and promising pattern recognition technique developed by Vapnik (1995) and his research group. The SVM-based shape recognition approach is realized by a multi-class classifier based on SVM. In this paper, a simple extension of one-against-one structure is used to construct the multi-class classifier. Comparing each class with another class pair-wise, and we get $n(n-1)/2$ classifiers in total. Each test sample will be classified by these classifiers.

Based on the max-win scheme (Mayoraz and Alpaydin 1999), each classifier casts one vote for its preferred class, and the final result is the class with the most votes. Compared with another traditional multi-class classifier construction structure—one-against-all, our approach has the following benefits: Firstly, the decision planes in one-against-all structure are always too complicated to avoid over-fitting. Secondly, in the one-against-all structure each classifier will have to train on the whole dataset. It is very time-consuming compared with the one-against-one structure.

In order to train our SVM classifiers, we collected 186 samples of different shape types drawn by several different users. Each sample is normalized to a 20x20 image. The pixel occupied by the sample stroke is marked with 1 and others with 0 in the image. 47 virtual samples are created for each sample by rotation and symmetrically mirroring. Hence, we obtained 8928 training samples in total. With the aids of virtual samples, the classifiers are made transformation-invariant to some extent. Each closed polyline (or polygon) is also normalized to a 20x20 image and input to the trained SVM classifiers. We chose the shape type with the maximum votes as the recognized shape.

4.3. The NN-Based Shape Recognition Approach

The neural network we used is quite generic, which is trained using the standard error back-propagation (BP) approach (Yaeger 1996).

The training set used for training the SVM classifiers is also used to train the neural network. The input layer of our neural network consists of 400 nodes, each corresponding to one pixel in the normalized image. There are 100 nodes and 40 nodes in the second layer and the third layer, separately. The numbers of the nodes of the second and third layers are determined based on experiments and experiences for the best output performance. The output (the fourth and last) layer has 3 nodes, standing for Triangle, Rectangle, and Ellipse, separately. Similarly to the SVM classifiers, each stroke is also classified by the neural network classifier. The shape type is determined as the shape with the biggest likeliness to the input polygon.

5. Parameter Estimation

Once the shape type is known, its parameters should be determined such that it is rendered and looks approximately like its sketchy shape to the best extent. Moreover, we also apply another two rules in parameter estimation: making the edges horizontal or vertical if they are nearly so and making the shape to the most regular shape (which is discussed in detail later) if it is nearly so. Hence, there are two steps in parameter estimation: fitting and regularization.

Since different shape type involves different parameters, we discuss parameter estimation for ellipses and multiple-edge shapes, separately. For ellipse fitting, we first determine its axes orientations by finding the eigenvectors of the covariance matrix of the sampling points along the stroke at equi-length steps. The axes are then rectified to horizontal or vertical according to a certain criterion. The stroke's bounding rectangle whose edges are parallel to the axes is used to determine the center and axes lengths of the ellipse. If the ratio of the lengths is nearly 1, it is rectified to a circle.

For N-edge polygon fitting, the problem is similar to a recursive polygonal approximation process until only N edges are left. We initialize the fitting polygon with the first N points (on the original polygon) with the smallest edge angles since they are most likely the final vertices of the fitting polygon. We then try to move each vertex of the fitting polygon to its neighbors on the original polygon and try to see if the area of the fitting polygon can increase. We finalize the fitting polygon's vertices by finding a fitting polygon with a relatively big area. The fitting polygon then divides the original polygon into N pieces. By finding the linear regression result of each piece, the edges of the fitted N-edge polygon can be obtained. After that, the edges are also rectified to horizontal or vertical as appropriate and the final vertices are determined by calculating the intersections of the rectified edges. Finally, the polygon is regularized to an equilateral triangle, isosceles triangle, right-angled triangle, scalene triangle, square, trapezoid, parallelogram, rhombuse, or a general quadrilateral as appropriate.

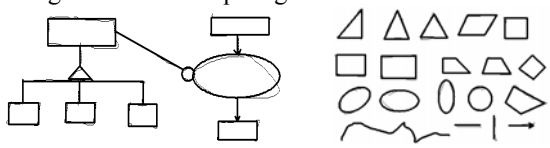
6. Compound Graphics Recognition

We can group several recently input graphic objects according to the sketching/drafting ways that people frequently use and see if some compound graphic object can be formed. The sketching method of an arrowhead is a good example, which usually consists of drawing a line first and then a wedge near one the line's endpoint. In this paper, compound graphic object recognition is rule-based. For instance, a wedged arrowhead is recognized if the wedge consists of two approximately equi-length segments forming an angle of about 45 degrees. A hollow arrowhead is recognized if a small triangle is drawn near an endpoint of an existing stroke. Since there are several sketching methods of an arrowhead, we should handle all of them. We will continue to add other classes of compound graphic objects into the Smart Sketchpad demo system in the future.

7. Implementation, Experiments, and Evaluation

Implementing the on-line graphics recognition

algorithm of the mentioned graphic objects in C++ on Microsoft Windows 2000, we develop a demo system called Smart Sketchpad. The original trajectory is displayed while the pen-tip is touching the tablet. Once the pen-tip is lifted up, the original sketchy line is erased and the recognized graphic object is displayed. The system also has the "UNDO" function. When "UNDO" is request, the recognized graphic object is erased and its original sketchy line is recovered. Currently, it can recognize 6 types of graphic objects: (various) triangles, (various) rectangles, (various) ellipses (/circles), arrowheads, straight lines, and free-form curves. More complex and powerful functions will be added in the future development of the system. Figure 2(a) shows a real diagram drawn using with the Smart Sketchpad system. Figure 2(b) shows some examples of single graphic object recognized by the Smart Sketchpad system. The original sketchy strokes are displayed in thin lines (for human vision evaluation) and the recognized graphics are displayed in thick lines. As we can see, the recognition effect is quite good.



(a) (b)

Figure 2. (a)Input a real life diagram using Smart Sketchpad.(b) Input some graphics.

In order to evaluate the system, we asked 5 different users to drawn three classes of closed shapes using the three shape-recognition approaches. The correct recognition rates are listed in Table 1. Our recognition precision by rule-based approach is very similar to others (Fonseca and Jorge 2000). ANN-based and SVM-based approaches can obtain better performances than rule-based approach.

Table 1. Performance evaluation

Approach	Shape	Total#	Correct#	Accuracy%
Rule-based	Triangle	160	137	85.6
	Rectangle	145	129	90.0
	Ellipse	187	187	100
	Total	492	453	92.1
ANN-based	Triangle	140	133	95.0
	Rectangle	108	106	98.1
	Ellipse	168	164	97.6
	Total	416	403	96.8
SVM-based	Triangle	195	190	97.4
	Rectangle	119	113	95.0
	Ellipse	171	170	99.4
	Total	485	473	97.5%

8. Concluding Remarks

An on-line graphics recognition algorithm is proposed and implemented in the Smart Sketchpad system, which currently can recognize free curves, straight lines, ellipses (circles), triangles (with different types: scalene, equilateral, isosceles, right-angled), rectangles (squares, general quadrilaterals, trapezoids, parallelograms, rhombuses), and arrowheads. Multiple stroke shapes can also be recognized. Other shapes will be added in the future. Experiments and evaluation are also presented, which show that ANN and SVM based approaches are more suitable for on-line graphic recognition problem. Good performance (97.5% by SVM) of the system is also shown. The system provides users a natural, convenient, and efficient way to input rigid and regular shape graphic objects by quickly drawing their sketchy shapes. We believe that by providing more recognition functions, the sketching system can increase the efficiency of drafting diagrams, graphs, and other kinds of drawings by a dramatic degree.

References

[1] Ajay A, Van V, and Takayuki DK (1993), "Recognizing Multistroke Geometric Shapes: An Experimental Evaluation", In: *Proc. of the 6th Annual ACM Symposium on User Interface Software and Technology*, pp. 121-128, 1993

[2] Albuquerque MP, Fonseca MJ, and Jorge JA (2000), "Visual Language for Sketching Document", In: *Proc. IEEE Sym. on Visual Languages*, pp. 225-232, 2000.

[3] Arvo J and Novins K (2000) Fluid Sketches: Continuous Recognition and Morphing of Simple Hand-Drawn Shapes. In: *Proc. of the 13th Annual ACM Symposium on User Interface Software and Technology*, San Diego, California, November, 2000.

[4] Fonseca MJ and Jorge JA (2000), "Using Fuzzy Logic to Recognize Geometric Shapes Interactively", In: *Proc. 9th IEEE Conf. on Fuzzy Systems*, Vol.1, pp. 291-296, 2000.

[5] Gross MD and Do EYL (1996) Ambiguous Intentions: A Paper-like Interface for Creative Design. In: *Proc. of the 9th Annual ACM Symposium on User Interface Software and Technology*, pp183-192, Seattle, WA, 1996

[6] Mayoraz E and Alpaydin E (1999) Support Vector Machines for Multiclass Classification. In: *Proc. of the Int. Workshop on Artificial Neural Networks*, 1999.

[7] Sklansky J and Gonzalez V (1980) Fast Polygonal Approximation of Digitized Curves. *Pattern Recognition* 12:327-331.

[8] Vapnik V (1995) *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

[9] Yaeger L (1996) Neural Networks Provide Robust Character Recognition for Newton PDAs. *IEEE Expert - Intelligent Systems and Their Applications* 11(4): 10-11.

[10] Zeleznik RC, Herndon KP, and Hughes JF (1996) SKETCH: An Interface for Sketching 3D Scences. In: *SIGGRAPH96*, pp. 163-170, New Orleans, August, 1996.