Recognition and Beautification of Multi-Stroke Symbols in Digital Ink

Heloise Hwawen Hse, A. Richard Newton

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720, U.S.A.
{hwawen, newton}@eecs.berkeley.edu

Abstract

Sketch-based user interfaces provide a more direct and convenient way for interacting with computers, especially for performing graphical tasks. Most computer programs provide a mouse-and-palette based user interface for editing shapes which can be cumbersome to use. In order to draw a shape, a user must first select the desired shape from a menu or from a hierarchy of menus, and then make a series of adjustments to the shape (i.e. rotation, scaling, horizontal/vertical flip, etc.). A more convenient approach to this task is to allow the user to sketch the desired shape directly and then replace it with a 'beautified' symbol with the correct transformation, all in one step. In this paper, we present a complete system for recognizing and beautifying sketched symbols. We have implemented this system as an interface to the Microsoft PowerPoint application to enable a user to sketch symbols directly onto a presentation slide.

Introduction

Sketching is a simple and natural mode of expression. It is especially desirable for conceptual design, both for personal work and in a collaborative environment. With a sketch-based user interface (UI), one can experience the freedom of sketching on paper and yet gain the benefits of an electronic design tool (Hearst et al. 1998). A sketch-based UI is especially suitable for communicating graphical ideas to computers (Igarashi 2003). The graphic utilities in most existing applications (e.g. Microsoft PowerPoint, Visio, Adobe Illustrator, etc.) are mouse-and-palette based, where a user is required to select a desired item from a tool palette or access it through a hierarchy of menus. This type of deep modal interaction is very controlling, because the user must tell the program both explicitly and precisely what to do. In fact, it is similar to typing commands in a command-line user interface, except the user is now specifying commands through buttons and menus (Igarashi 2003). The advantage is that the interaction leaves no room for ambiguity and therefore less likely to produce errors, but the disadvantage is that the process can be cumbersome and tedious for the user. For example, in order to draw a simple parallelogram in PowerPoint 2003, like the one in Figure 1, one would need to perform the following steps:

1. select *AutoShapes* menu from the *Drawing* toolbar
2. select *Basic Shapes* menu
3. click on *parallelogram*
4. click and drag the shape to the desired size on the slide
5. flip the parallelogram either horizontally or vertically

Once these tasks are completed, one may still need to reposition the shape. If rotation is desired, an additional step is also required.

Figure 1. A parallelogram created in PowerPoint using a menu-based interface with a horizontal flip applied.

If the user can sketch the shape directly the way he or she would like it to appear, and the sketched shape can be 'neatened' into the corresponding PowerPoint or other application-specific object, the procedure of creating a shape would be far more straightforward, just like drawing with pen and paper. With such an interface, not only is the simplicity of pen and paper preserved, but also the resulting sketch can be interpreted with full and direct knowledge of intent, thus realizing the full benefit of an electronic tool. Of course, the ink can also be left unformatted to facilitate further design exploration if desired (Landay et al. 2002).

In this paper, we describe a collection of algorithms we have developed that facilitate a pen-and-paper interface for creating graphic symbols. We have developed a system, *HHreco*, which recognizes and beautifies a sketched symbol (Hse 2004). This system can be integrated easily into an existing application for application-specific beautification. We have chosen PowerPoint as our initial target application and have interfaced *HHreco* to PowerPoint 2003 to create a sketch-based user interface for sketching and beautifying symbols on a presentation slide.

System Architecture

The steps that take a sketched shape to its beautified version include recognition, fragmentation, and beautification (see Figure 2). More specifically, once a user has sketched a symbol using one or more pen strokes, the strokes are passed to the recognition engine to be recognized. The symbol is then structurally decomposed into its primitive elements, line segments and elliptical arcs. Based on the prediction of the recognizer, the symbol can be accurately decomposed using a template based approach. From the structural elements, geometric properties are computed to derive the adjustment parameters for appropriate beautification in PowerPoint.

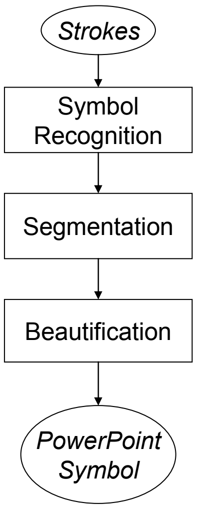

Figure 2. The overall flow of the symbol recognition and beautification system.

Data Acquisition

The data acquisition process has been presented in (Hse and Newton 2004) and is summarized here for completeness. Since there are no publicly available benchmark for sketched symbols, we have created a symbol database by gathering data from different people. The target shapes were chosen based upon the applications of interest (e.g. a slide drawing program like Microsoft PowerPoint, a UML diagram editor, or an electrical schematic editing tool) and include often-used basic shapes and shapes with different geometric structures, including shapes with lines, with curves, with mixed lines and curves, and shapes with and without self-intersection. Of course, other shapes can be added and learned by the system, if desired.

To date, we have gathered data from 19 people. Each participant was asked to sketch 30 examples for each of the 13 symbols shown in Figure 3. The database contains a total of 7,410 examples overall and 570 examples per symbol. The data was collected using the Wacom Graphire2 Pen and Tablet.

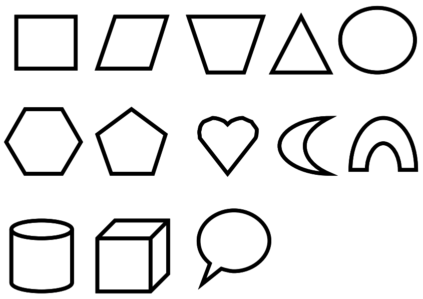

Figure 3. The target symbol set.

Symbol Recognition

In order for users of a system to sketch naturally, the recognition system should not place constraints on how the user can draw a symbol in order for it to be recognized accurately. The user should be able to draw a shape in the same way he or she would on a piece of paper without having to worry about where to start a stroke, how many strokes to use, in what order to draw the strokes, etc. For the class of applications we are interested in, the recognition should be independent of stroke-order, -number, and -direction, as well as invariant to rotation, scaling, translation, and reflection of symbols. Based on these considerations, we evaluated various types of moments which have been used in the image recognition community and a variety of different machine learning techniques. We selected Zernike moments as the key abstraction for the feature descriptors of symbols for three main reasons: they have been shown effective in image representation (Teh and Chin 1988), the magnitudes of the Zernike moments are invariant to rotation and reflection (Khotanzad and Hong 1990; Bailey and Srinath 1996), and they can be constructed easily to an arbitrary order, facilitating the construction of the feature set (Teh and Chin 1988). Through our experiments, we found that using a feature set consisting of feature values ranging from moment order 2 and up to order 8 is sufficient for the recognition problem. Using higher-order moments does not yield significant improvement in the recognition rate (<1%); in some cases the accuracy rate decreases since higher order moments are more susceptible to noise (Teh and Chin 1988). With a Support Vector Machine (SVM) classifier using a radial basis kernel (Vapnik 1995), the recognition rate reaches 97% in both writer-dependent and writer-independent tests. More detailed descriptions of the method and the experimental results have been published elsewhere (Hse and Newton 2004).

Symbol Segmentation

Once a sketched symbol has been recognized, structural information is needed to determine the geometric properties of the symbol in order to perform meaningful beautification. For example, the orientation of the symbol, the width and height of the symbol, the sides that are parallel or perpendicular, the sides that are equal in length, etc. This structural information is obtained by segmenting the symbol into a set of basis elements. In our system, the basis set consists of straight line segments (L) and elliptical arcs (E) (Hse, Shilman, and Newton 2004). They are simple and minimal bases that we have found to best represent the family of symbols considered in this work.

Based on the recognition result for a sketched symbol, the appropriate segmentation template can be determined. A template description consists of the number of E's and the number of L's. The segmentation problem is set up as an optimization problem in which the cost function is the fit error and the constraint is the symbol template. Based upon the template description of a symbol, the algorithm identifies the globally optimal set of breakpoints that minimizes the fit error for the symbol. Using a dynamic programming approach, the optimal solution can be obtained in polynomial time (Hse, Shilman, and Newton 2004).

This algorithm is very robust in determining breakpoints that closely match a human's natural perception. In addition to finding the breakpoints, the fitted segments produced by the algorithm can serve as the basis for beautification of a symbol. The symbol can be further beautified based on the geometric properties derived from the segments, and this procedure is elaborated in the next section. For complex symbols with mixed line segments and elliptical arcs such as arches, cylinders, and callouts, the strokes are filtered to reduce the number of data points in order to expedite the segmentation process.

Symbol Beautification

For symbols that have been recognized as polygons, the sketched strokes are ordered by endpoint proximity right before the segmentation routine is called. The ordering of the sketched strokes allows the resulting fitted segments to be arranged in a cyclic order which eases the processing and the analysis steps that are to follow. To instantiate one of the PowerPoint shapes in Figure 3, four parameters must be specified, namely the *x* and *y* coordinates of the upper-left corner of the shape, and the width and height of the shape. The shape can only be created initially in the upright position. Depending on the sketched symbol, additional features such as rotation, flip, and other adjustments will have to be applied after the symbol has been instantiated. A rotation must be performed with respect to the center of the shape's bounding box so as to avoid shifting out of the correct position after the rotation.

The upper, left corner that is needed to instantiate a symbol is not necessarily the upper, left corner of the sketched symbol's bounding box because the sketched symbol may not be in an upright orientation. The bounding box from the PowerPoint ink shape is the smallest upright rectangle that encloses the ink strokes, but this bounding box is not the "desired" bounding box adapted to the orientation of the shape (see Figure 4). Therefore, the upper, left corner for instantiating a new symbol has to be derived carefully from the fitted segments such that after the rotation transform, the symbol will align with the sketched shape.

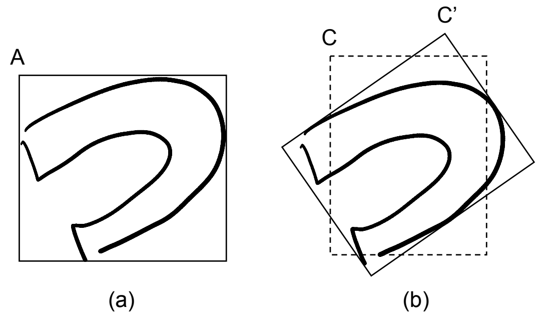

Figure 4. A sketched arch. (a) The bounding box given by the PowerPoint ink object. (b) The desired bounding box is shown with a thin solid line. "C" is the coordinate needed to instantiate the PowerPoint symbol to ensure the correct alignment after the rotation transformation has been applied.

The extent to which beautification can be performed is in part limited by the available support of an application. For example, it is not possible to create an asymmetric trapezoid in PowerPoint, and therefore a sketched asymmetric trapezoid will be beautified into the closest symmetric trapezoid.

A crucial procedure in the beautification process is to identify the correct segment correspondence between a PowerPoint symbol and a sketched symbol because this directly affects the correctness of the transformation parameters calculated. The challenging factor lies in that a sketched symbol is not always drawn in the upright position, so the alignment of the segments in a sketched symbol and a PowerPoint symbol has to be derived carefully. The transformation parameters are calculated from the fitting segments instead of the original strokes because the fitting segments tend to carry more stable information, whereas sketched strokes contain more noise and irregularities.

In Figures 6-18, we refer to the pointing end of a segment as the head, H, and the other end as the tail, T (see Figure 5).

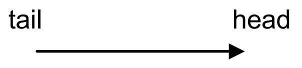

Figure 5. The head and tail of a directed segment.

Rectangles Since the line segments have been ordered, it is safe to assume that the first and the third segments are parallel, and the second and the fourth segments are parallel. We computed the width of the rectangle from segments 1 and 3, and the height from segments 2 and 4. Of course it could be the other way around, however the rotation transform would be different. The object is snapped to the closest axis (vertical or horizontal) if the orientation is within ±10º of the axis (Figure 6).

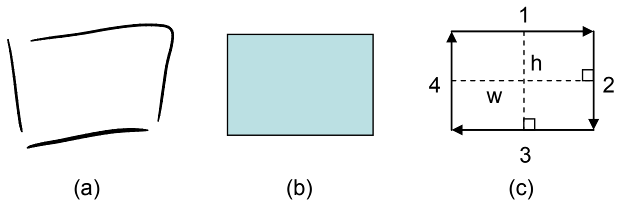

Figure 6. (a) Sketched rectangle. (b) Beautified rectangle. (c) Graphical description of the parameters.

Parallelograms The line segments are ordered consecutively and the traversal starts with a segment that forms an acute angle with the next segment. This specific arrangement makes it easier to calculate the degree of rotation and whether or not an instantiated PowerPoint parallelogram must be flipped. If the traversal is counterclockwise, then a horizontal flip is required (see Figure 7). The averaged length of segments 1 and 3 is used to adjust the shear parameter of the parallelogram.

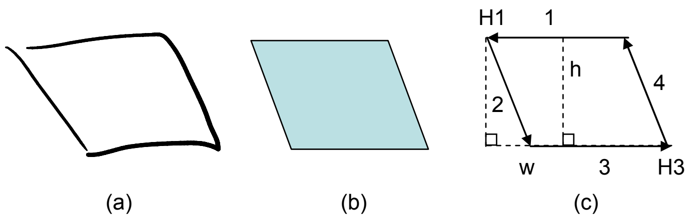

Figure 7. (a) Sketched parallelogram. (b) Beautified parallelogram. (c) Graphical description of the parameters.

Trapezoids The pair of parallel sides are determined by computing the dot products of the opposing segments. Next, the top (short side) and the bottom (long side) of the trapezoid are identified and then the segments are arranged in clockwise order starting from the top segment. Then the rotation angle is computed and snapped to the closest horizontal or vertical axis if the angle is within ±10º of the axis. (see Figure 8)

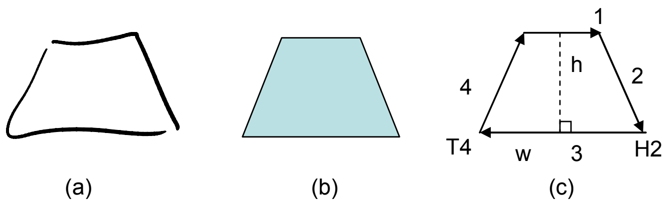

Figure 8. (a) Sketched trapezoid. (b) Beautified trapezoid. (c) Graphical description of the parameters.

Triangles Even though only two types of triangles are supported directly in PowerPoint, right triangles and isosceles triangles, our system supports right triangle, isosceles triangles, equilateral triangles, and other triangles as well. If one of the angles in a sketched triangle is close to 90º (±10º), it is beautified into a right triangle. Then the line segments are ordered clockwise starting from the base segment going toward the right angle (see Figure 9). If two sides of a sketched triangle are close in length, it is beautified into an isosceles triangle. Starting from the base of the isosceles triangle, the line segments are ordered clockwise. Of course if an isosceles triangle contains a right angle, it will be beautified accordingly. The width, height, and the rotation of the triangle can then be easily computed with the base segment anchored. If all three sides of the triangles are about equal, it is beautified into an equilateral triangle. A triangle that does not fall into any of these special categories is handled by fixing the longest side as the base, and then creating a PowerPoint isosceles triangle and adjusting its shear parameter to fit the sketched triangle.

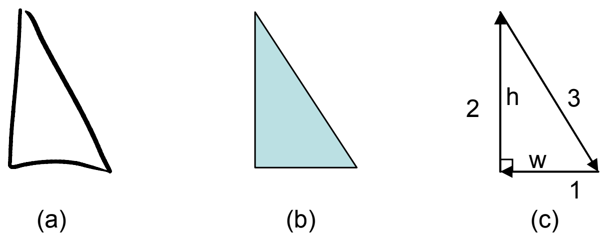

Figure 9. (a) Sketched triangle with a right angle. (b) Beautified right triangle. (c) Right triangle parameters.

Regular Pentagons The width of a pentagon is the distance between H1 and T5 (see Figure 10). To determine the rotation angle of a pentagon, first, the intersection (P) between the first and the last line segments of the pentagon is calculated. Next, a vector (V) going from the center of the pentagon to P is formed. Then the rotation angle is determined from the pointing direction of V.

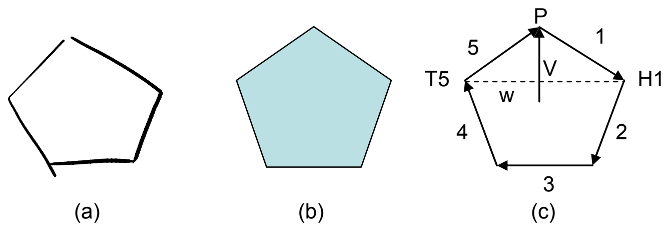

Figure 10. (a) Sketched pentagon. (b) Beautified pentagon. (c) Graphical description of the parameters.

Regular Hexagons Since the fitted line segments of a sketched hexagon have been ordered consecutively, the width of the hexagon is calculated by taking the distance between H2 and T6. The rotation vector (V) is determined by the intersection of segments 2 and 6 and the center of the hexagon. (see Figure 11)

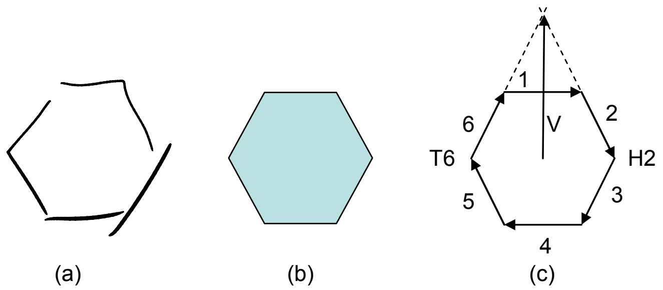

Figure 11. (a) Sketched hexagon with fitted line segments ordered consecutively. (b) Beautified hexagon. (c) Graphical description of the parameters.

Ellipses From the elliptical fitting, the major and minor axes and their magnitudes are computed. Using the major axis, the orientation of the sketched ellipse is determined, and the appropriate rotation angle is applied to a PowerPoint elliptical symbol. (see Figure 12)

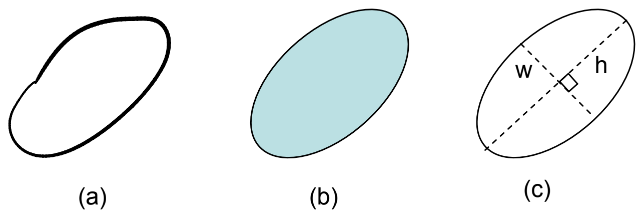

Figure 12. (a) Sketched ellipse. (b) Beautified ellipse. (c) Graphical description of the parameters.

Hearts The rotation angle of a heart shape is calculated from the major axes of the two fitting elliptical arcs. First, the intersection (P1) of the two major axes is computed. Next, the two ends of the major axes that are closer to the bottom of the hearts are identified. Then, the midpoint (P2) of the segment formed by these two endpoints is calculated. The vector going from P2 to P1 is used to determine the rotation of the heart shape. (see Figure 13)

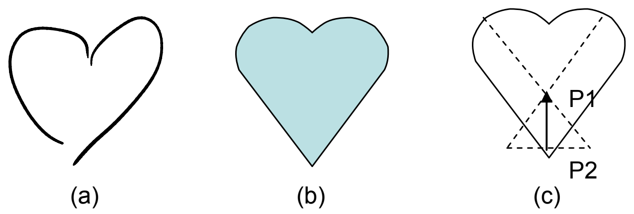

Figure 13. (a) Sketched heart. (b) Beautified heart. (c) Graphical description of the parameters.

Moons The rotation angle of the moon is determined by the vector that points toward the concave opening of the moon. This vector (V) is obtained by taking the midpoint of the outer elliptical arc and the midpoint of the segment formed by the arc's endpoints. The magnitude of this vector is the width of the moon. To adjust the thickness parameter, the distance (D) between the midpoints of the two elliptical arcs is computed, and then the ratio of D and the width of the moon are calculated. (see Figure 14)

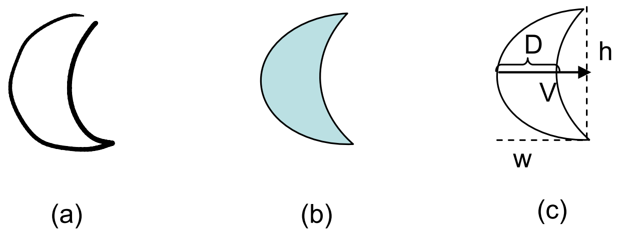

Figure 14. (a) Sketched moon. (b) Beautified moon. (c) Graphical description of the parameters.

Arches (Block arcs) The rotation angle is determined by the vector (V) from the midpoint of the outer elliptical arc to the midpoint of the segment formed by the endpoints of the inner elliptical arc. The height to create the PowerPoint arch symbol is twice the magnitude of V, and the width of the arch is the length of the segment formed by the endpoints of the outer elliptical arc. (see Figure 15)

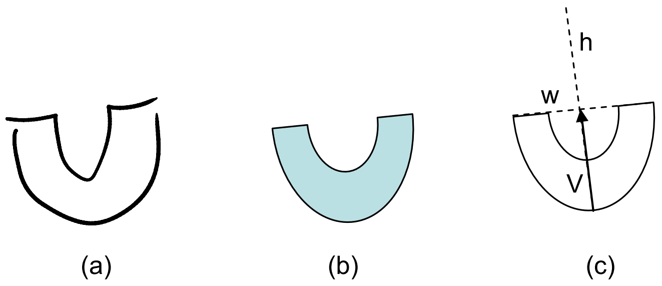

Figure 15. (a) Sketched arch. (b) Beautified arch. (c) Graphical description of the parameters.

Cylinders The orientation of a cylinder symbol is determined by the vector (V) from the middle of the cylinder (midway between the L1 and L2) to the center of the ellipse E1. The height of the cylinder is the distance from the midpoint of E2 along the orientation vector to the top of E1. The intersections of the orientation vector and

E1 are computed. Two possible intersections exist and the correct one would be the intersection point that yields a greater height distance. (see Figure 16)

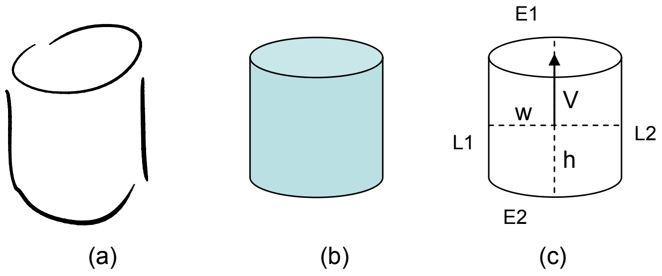

Figure 16. (a) Sketched cylinder. (b) Beautified cylinder. (c) Graphical description of the parameters.

Cubes The current implementation assumes that cubes are sketched in the upright position. The three horizontal line segments in a cube are identified and their relative positions are used to determine whether an instantiated PowerPoint cube needs to be horizontally flipped. (see Figure 17)

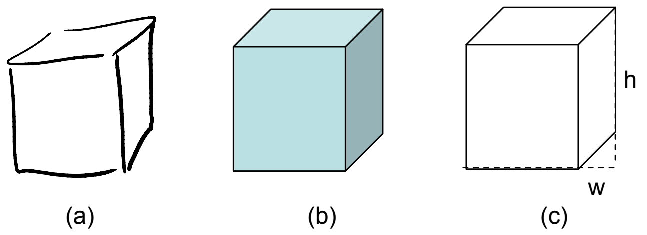

Figure 17. (a) Sketched cube. (b) Beautified cube. (c) Graphical description of the parameters.

Callouts A callout symbol is fitted with one ellipse and two lines. The magnitudes of the major and minor axes of the fitted ellipse determine the width and height of the callout symbol, and the orientation of the major axis indicates the rotation angle. Once the body of the callout symbol has been decided, the pointing direction is obtained by taking the intersection of the two line segments. The tip of the pointer has to be specified in terms of the relative position to the width and height of the symbol body. (see Figure 18)

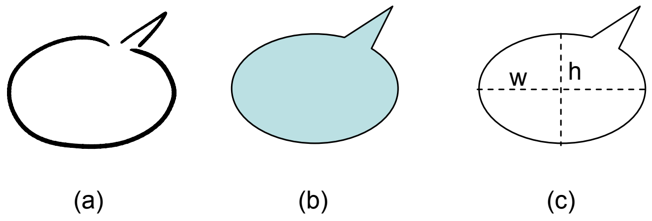

Figure 18. (a) Sketched callout. (b) Beautified callout. (c) Graphical description of the parameters.

System Implementation

The recognition system and the symbol segmentation utilities have been implemented in Java and released in a software package, *HHreco*, that is available for download (Hse 2004). In order to interface to the PowerPoint 2003 application, program development was carried out in Visual Studio .NET 2003. The *HHreco* library was ported over to J# in .NET. Since the beautification routines are application-specific, they are implemented in C# in order to better interface to the PowerPoint object model.

At program start-up, a PowerPoint application and a recognition toolbar are launched. The recognition toolbar (Figure 19) contains a button to invoke recognition and a menu for correcting misrecognitions. By selecting the pen tool on the *Ink Drawing and Writing* toolbar in PowerPoint, one can begin to sketch shapes on a presentation slide. The inks are directly accepted by the PowerPoint application. Once a symbol has been completed (Figure 20), the user clicks on the *Recognize ink* button to invoke the recognizer. Since PowerPoint does not expose ink data through its object model at this time, all of the processing must be carried out externally. The ink object is copied from the PowerPoint application to a clipboard and pasted into our own component for recognition, segmentation, and beautification. Once the transformation parameters have been computed, a properly transformed PowerPoint symbol is then created and displayed on the slide (Figure 21). When a misrecognition occurs, the user can make correction by selecting the correct class from the drop down menu. This action will incrementally retrain the recognizer with the sketched symbol and its correct class label, and then reprocess the ink and update the display with the correct symbol. The adaptability of the recognition system allows it to continue to improve its accuracy by learning from users' corrections.

At the current time, we are unable to programmatically delete the inking strokes in the PowerPoint slide due to accessibility issues in the PowerPoint 2003 API (at this time, the ink collector of PowerPoint is not exposed through the object model).

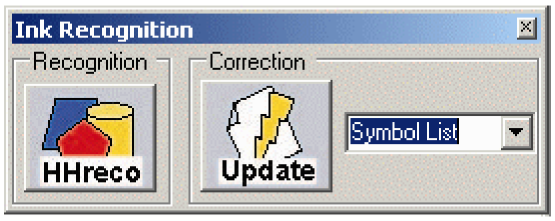

Figure 19. The recognition toolbar contains a button to call the recognizer and a correction interface.

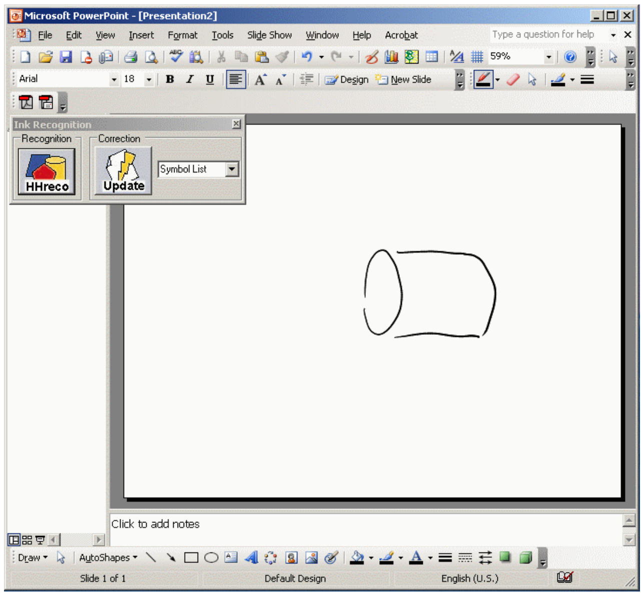

Figure 20. A sketched cylinder.

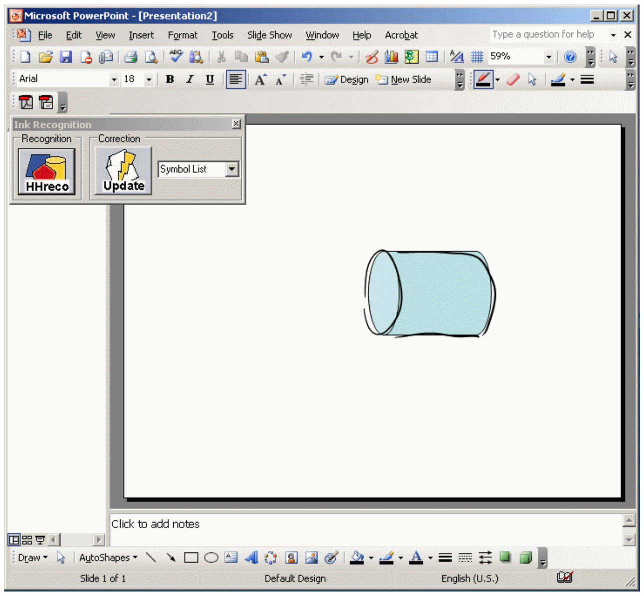

Figure 21. Cylinder beautified into a PowerPoint object overlaying the original ink strokes.

Conclusions and Future Work

We have developed a complete system for recognizing and beautifying hand-sketched symbols. The multi-stroke recognition system is independent of stroke order, number, and direction, as well as being invariant to rotation, scaling, translation, and reflection of symbols. Furthermore, the system is trainable and adaptive such that it is capable of learning new shapes from examples and improving its accuracy over time. The robust symbol segmentation algorithm allows structural primitives to be extracted with high accuracy in order to facilitate the beautification process. We have integrated this system with the PowerPoint 2003 application to enable sketching graphic symbols directly onto a presentation slide.

In informal tests, users have expressed positive feedback on the sketch-based PowerPoint interface. They feel that this is a more straight-forward and convenient way of entering symbols than through the conventional menus. Our future work includes a more in-depth user study and exploring the possibilities of using a timeout scheme or scene segmentation to automatically group strokes and invoke recognition.

Acknowledgement

We would like to thank Michael Shilman and Barn Li of Microsoft for their technical advice on interfacing to PowerPoint. This work was supported in part by the Roy W. Carlson Distinguished Professorship. This support is gratefully acknowledged.

References

Bailey, R. and Srinath, M. 1996. Orthogonal Moment Features for Use with Parametric and Non-Parametric Classifiers. *IEEE Trans. on PAMI* 18(4): 389-399.
Hearst, M. A., Gross, M. D., Landay, J. A. and Stahovich, T. F. 1998. Sketching Intelligent Systems. *IEEE Intelligent System* 3(3): 10-19.
Hse, H. 2004. HHreco. http://www-cad.eecs.berkeley.edu/Respep/Research/hhreco/index.html
Hse, H. and Newton, A. R. 2004. Sketched Symbol Recognition using Zernike Moments. *2004 International Conference on Pattern Recognition*. Forthcoming.
Hse, H., Shilman, M. and Newton, A. R. 2004. Robust Sketched Symbol Fragmentation using Templates. *International Conference on Intelligent User Interfaces*,156-160. Funchal, Portugal, ACM Press.
Igarashi, T. 2003. Freeform User Interfaces for Graphical Computing. *International Symposium on Smart Graphics*,39-48. Heidelberg, Germany.
Khotanzad, A. and Hong, Y. H. 1990. Invariant Image Recognition by Zernike Moments. *IEEE Trans. on PAMI* 12(5): 289-297.
Landay, J. A., Hong, J. I., Klemmer, S., Lin, J. and Newman, M. 2002. Informal PUIs: No Recognition Required. *AAAI Spring Symposium on Sketch Understanding*,86-90. Palo Alto, CA, AAAI Press.
Teh, C. and Chin, R. T. 1988. On Image Analysis by the Methods of Moments. *IEEE Trans. on PAMI* 10(4): 496-513.
Vapnik, V. 1995. *The Nature of Statistical Learning Theory*. New York, Springer-Verlag.