An Image-Based Trainable Symbol Recognizer for Sketch-Based Interfaces

Levent Burak Kara
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
lkara@andrew.cmu.edu

Thomas F. Stahovich
Mechanical Engineering Department
University of California, Riverside
Riverside, California 92521
stahov@engr.ucr.edu

Abstract

We describe a trainable, hand-drawn symbol recognizer based on a multi-layer recognition scheme. Symbols are internally represented as binary templates. An ensemble of four template classifiers ranks each definition according to similarity with an unknown symbol. Scores from the individual classifiers are then aggregated to determine the best definition for the unknown. Ordinarily, template-matching is sensitive to rotation, and existing solutions for rotation invariance are too expensive for interactive use. We have developed an efficient technique for achieving rotation invariance based on polar coordinates. This techniques also filters out the bulk of unlikely definitions, thereby simplifying the task of the multi-classifier recognition step.

Introduction

A long standing challenge in pen-based interaction concerns *symbol recognition*, the task of recognizing individual hand-drawn figures such as geometric shapes, glyphs and symbols. While there has been significant recent progress in symbol recognition (Rubine 1991; Fonseca, Pimentel, & Jorge 2002; Matsakis 1999; Hammond & Davis 2003), many recognizers are either hard-coded or require large sets of training data to reliably learn new symbol definitions. Such issues make it difficult to extend these systems to new domains with novel shapes and symbols. The work presented here is focused on the development of a trainable symbol recognizer that provides (1) interactive performance, (2) easy extensibility to new shapes, and (3) fast training capabilities.

Our recognizer uses an image-based recognition approach. This approach has a number of desirable characteristics. First, segmentation – the process of decomposing the sketch into constituent primitives such as lines and curves – is eliminated entirely. Second, our system is well suited for recognizing "sketchy" symbols such as those shown in Figure 1. Lastly, multiple pen strokes or different drawing orders do not pose difficulty. Many of the existing recognition approaches have either relied on single stroke methods in which an entire symbol must be drawn in a single pen stroke (Rubine 1991; Kimura, Apte, & Sengupta 1994), or constant drawing order methods in which two similarly shaped patterns are considered different unless the pen strokes leading to those shapes follow the same sequence (Ozer *et al.* 2001; Yasuda, Takahashi, & Matsumoto 2000).

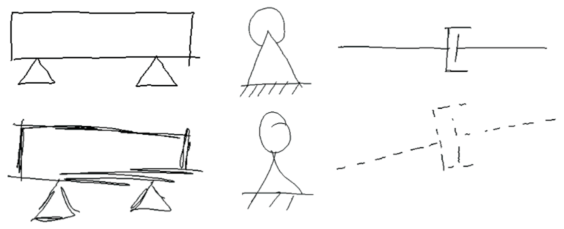


Figure 1: Examples of symbols correctly recognized by our system. The top row shows symbols used in training and the bottom row shows correctly recognized test symbols. At the time of the test, the database contained 104 definition symbols.

Unlike many traditional methods, our shape recognizer can learn new symbol definitions from a *single* prototype example. Because only one example is needed, users can seamlessly train new symbols, and remove or overwrite existing ones on the fly, without having to depart the main application. This makes it easy for users to extend and customize their symbol libraries. To increase the flexibility of a definition, the user can provide additional examples of a symbol.

Ordinarily, template-matching is sensitive to rotation, and existing solutions for rotation invariance are too expensive for interactive use. We have developed an efficient technique for rotation invariance based on a novel polar coordinate analysis. The unknown symbol is transformed into a polar coordinate representation, which allows the program to efficiently determine which orientation of the unknown best matches a given definition. During this process, definitions that are found to be markedly dissimilar to the unknown are pruned away, and the remaining ones are kept for further analysis. In a second step, recognition switches to screen coordinates where the surviving definitions are analyzed in more detail using an ensemble of four different classifiers. Each classifier produces a list of definitions ranked according to their similarity to the unknown. In the final step of recognition, results of the individual classifiers are pooled together to produce the recognizer's final decision.

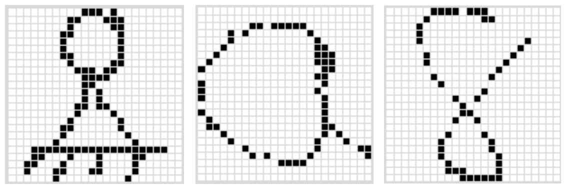


Figure 2: Examples of symbol templates: A mechanical pivot, letter ‘a’, digit ‘8’. The examples are demonstrated on 24x24 templates to better illustrate the quantization.

The analysis in polar coordinates precedes the analysis in screen coordinates. However, for the sake of presentation, we have found it useful to begin the discussion with our template representation and the four template matching techniques, since some of those concepts are necessary to set the context for the analysis in polar coordinates.

Template Matching

Symbols are drawn using a 9 x 12 Wacom Intuos2 digitizing tablet and a cordless stylus. Data points are collected as time sequenced (x,y) coordinates sampled along the stylus' trajectory. There is no restriction on the number of strokes, and symbols can be drawn anywhere on the tablet, in any size and orientation.

Input symbols are internally described as 48x48 quantized bitmap images which we call “templates” (Figure 2). This quantization significantly reduces the amount of data to consider while preserving the patterns’ distinguishing characteristics. The template representation preserves the original aspect ratio so that one can distinguish between, say, a circle and an ellipse.

During recognition, the template of the unknown is matched against the templates in the database of definitions. We use four different methods to evaluate the match between a pair of templates. The first two methods are based on the Hausdorff distance, which measures the dissimilarity between two point sets. Hausdorff-based methods have been successfully applied to object detection in complex scenes (Rucklidge 1996; Sim, Kwon, & Park 1999), but only a few researchers have recently employed them for hand-drawn pattern recognition (Cheung, Yeung, & Chin 2002; Miller, Matsakis, & Viola 2000). Our other two recognition methods are based on the Tanimoto and Yule coefficients. The Tanimoto coefficient is extensively used in chemical informatics such as drug testing, where the goal is to identify an unknown molecular structure by matching it against known structures in a database (Flower 1998). The Yule coefficient has been proposed as a robust measure for binary template matching (Tubbs 1989). To the best of our knowledge, the Tanimoto and Yule measures have not previously been applied to handwritten pattern recognition. In the following paragraphs we detail these four classification methods.

Hausdorff Distance

The Hausdorff distance between two point sets A and B is defined as:

$$H(A,B) = max(h(A,B), h(B,A))$$

where

$$h(A,B) = \underset{a\in A}{max}(\underset{b\in B}{min}\,\|a-b\|)$$

$\|a-b\|$ represents a measure of distance (*e.g.*, the Euclidian distance) between two points a and b. $h(A,B)$ is referred to as the directed Hausdorff distance from A to B and corresponds to the maximum of all the distances one can measure from each point in A to the closest point in B. The intuitive idea is that if $h(A,B) = d$, then every point in set A is at most distance d away from some point in B. $h(B,A)$ is the directed distance from B to A and is computed in a similar way. Note that in general $h(A,B) \neq h(B,A)$. The Hausdorff distance is defined as the maximum of the two directed distances.

In its original form, the Hausdorff distance is too sensitive to outliers. The *Partial* Hausdorff distance proposed by Rucklidge (1996) eliminates this problem by ranking the points in A according to their distances to points in B in descending order, and assigning the distance of the k^{th} ranked point as $h(A,B)$. The partial Hausdorff distance from A to B is thus given by:

$$h^k(A,B) = \underset{a\in A}{k^{th}}\,\underset{b\in B}{min}\,\|a-b\|$$

The partial Hausdorff distance, in effect, softens the distance measure by discarding points that are maximally far away from the counterpart point set. The results reported in the following sections are based on a rank of 6%, *i.e.*, in the calculation of the directed distances, the most distant 6% of the points are ignored. We determined this cutoff value empirically based on the user experience with our system.

Modified Hausdorff Distance

Modified Hausdorff Distance (MHD) (Dubuisson & Jain 1994) replaces the max operator in the directed distance calculation by the average of the distances:

$$h_{mod}(A,B) = \frac{1}{N_a}\sum_{a\in A}\underset{b\in B}{min}\,\|a-b\|$$

where N_a is the number of points in A. The modified Hausdorff distance is then defined as the maximum of the two directed average distances:

$$MHD(A,B) = max(h_{mod}(A,B), h_{mod}(B,A))$$

Although $h_{mod}(A,B)$ may appear similar to $h^k(A,B)$ with $k = 50\%$, the difference is that the former corresponds to the mean directed distance while the latter corresponds to the median. Dubuisson and Jain argue that for object matching purposes, the average directed distance is more reliable than the partial directed distance mainly because as the noise level increases, the former degrades gracefully whereas the latter exhibits a pass/no-pass behavior.

Tanimoto Similarity Coefficient

The Tanimoto Similarity coefficient (Fligner *et al.* 2001) between two binary images A and B is defined as:

$$T_{sc}(A,B) = \alpha \cdot T(A,B) + (1-\alpha) \cdot T^C(A,B)$$

where $T(A,B)$ and $T^C(A,B)$ are the Tanimoto coefficient and the Tanimoto coefficient complement, respectively. $T(A,B)$ is a measure of matching black pixels and is defined as:

$$T(A,B) = \frac{n_{ab}}{n_a + n_b - n_{ab}}$$

where n_a and n_b are the total number of black pixels in A and B respectively. n_{ab} is the number of overlapping black pixels. $T^C(A,B)$ is defined in a similar way except it takes into account the number of matching white pixels as opposed to the matching black pixels. α is a weighting factor that controls the relative contributions of $T(A,B)$ and $T^C(A,B)$. We typically set the value of α in the range [0.5,0.75]. This choice is justified by the fact that hand-drawn symbols usually consist of thin lines (unless excessive over-tracing is done), which makes the match of black pixels more informative than the match of white pixels. Hence, for our problem, the Tanimoto Similarity coefficient should be controlled more by $T(A,B)$ than by $T^C(A,B)$.

Similarity measures that are based exclusively on the number of overlapping pixels, such as the Tanimoto coefficient, often suffer from slight misalignments of the rasterized images. We have found this problem to be particularly severe for hand-drawn patterns where rasterized images of ostensibly similar shapes are almost always disparate, either due to differences in shape, or more subtly, due to differences in drawing dynamics. The latter commonly occurs as a result of irregular drawing speed, often manifesting itself as unevenly sampled digital ink. Hence, for two shapes drawn at different speeds, the resulting rasterized images will likely exhibit differences. In order to absorb such variations during matching, we use a thresholded matching criterion that considers two pixels to be overlapping if they are separated by a distance less than 1/15th of the image's diagonal length. For a 48x48 image grid, this translates into 4.5 pixels, *i.e.*, two points are considered to be overlapping if the distance between them is less than 4.5 pixels.

Yule Coefficient

The Yule coefficient, also known as the coefficient of colligation, is defined as:

$$Y(A,B) = \frac{n_{ab} \cdot n_{00} - (n_a - n_{ab}) \cdot (n_b - n_{ab})}{n_{ab} \cdot n_{00} + (n_a - n_{ab}) \cdot (n_b - n_{ab})}$$

where the term $(n_a - n_{ab})$ corresponds to the number of black pixels in A that do not have a match in B. Similarly, $(n_b - n_{ab})$ is the number of black pixels in B that do not find a match in A.

$Y(A,B)$ produces values between 1.0 (maximum similarity) and -1.0 (minimum similarity). Like the Tanimoto coefficient, the Yule coefficient it is sensitive to slight misalignments between patterns for the reasons explained above. A thresholded matching criterion is thus employed, which is similar to the one we use with the Tanimoto method.

Tubbs (1989) originally employed this measure for generic, noise-free binary template matching problems. By using a threshold, we have made the technique useful when there is considerable noise, as is the case with hand-drawn shapes.

Combining Classifiers

Our recognizer compares the unknown symbol to each of the definitions using the four classifiers explained above. The next step in recognition is to identify the true class of the unknown by synthesizing the results of the component classifiers. However, the outputs of the classifiers are not compatible in their original forms because: (1) The first two classifiers are measures of *dissimilarity* while the last two are measures of *similarity*, and (2) the classifiers have dissimilar ranges. To establish a congruent ranking scheme, we first transform the Tanimoto and Yule similarity coefficients into distance measures and then normalize the values of all four classifiers to the range 0 to 1. We refer to these two processes as parallelization and normalization.

Parallelization: To facilitate discussion, let M denote the number of definitions, R denote the number of classifiers and ${d_m}^r$ denote the score classifier r assigns to definition m. In our case $r \in \{$Hausdorff, Modified Hausdorff, Tanimoto, Yule$\}$ and m is any definition symbol in the database. We transform the Tanimoto and Yule coefficients into dissimilarity measures by reversing their values as follows:

For $m = 1, ..M$,

$${d_m}^{Tanimoto} \leftarrow 1.0 - {d_m}^{Tanimoto}$$

$${d_m}^{Yule} \leftarrow 1.0 - {d_m}^{Yule}$$

This process brings the Tanimoto and Yule coefficients in parallel with the Hausdorff measures in the sense that the numerical scores of all classifiers now increase with increasing dissimilarity.

Normalization: After parallelization, all classifiers become measures of distance but still remain incompatible due to differences in their ranges. To establish a unified scale among classifiers, we use a linear transformation function that converts the original distances into normalized distances. For this, we first find the smallest and largest values observed for each of the four classifiers:

$$minscore^r = \overset{M}{\underset{k=1}{min}}\ {d_k}^r \quad , maxscore^r = \overset{M}{\underset{k=1}{max}}\ {d_k}^r$$

The normalized distance $\bar{d_m}^r$ for definition m under classifier r is then defined as:

$$\bar{d_m}^r = \frac{{d_m}^r - minscore^r}{maxscore^r - minscore^r}$$

This transformation maps the distance scores of each classifier to the range [0,1] while preserving the relative order established by that classifier.

Combination Rule: Having standardized the outputs of the four classifiers by parallelization and normalization, we are now ready to combine the results. We use an approach similar to the sum rule introduced by Kittler *et al.* (1998). For each definition m, we define a combined normalized distance D_m by summing the normalized distances computed by the constituent classifiers:

$$D_m = \sum_{r=1}^{R} \bar{d_m}^{r}$$

Finally, the unknown pattern is assigned to the class having the minimum combined normalized distance. The decision rule is thus:

Assign unknown symbol to definition m^* if

$$m^* = \underset{m}{argmin}\, D_m$$

Handling Rotations

Template matching techniques are sensitive to orientation. Therefore, for rotation invariant recognition, it is necessary to first rotate the patterns into the same orientation. Often this is accomplished by incrementally rotating one pattern relative to the other until the best alignment is achieved. However, this is overwhelmingly expensive for real-time applications due to the costly rotation operation. We have developed a technique, based on the polar coordinate transformation, to greatly facilitate this process. The main idea is that rotations in Cartesian coordinates become translations in polar coordinates. Figure 3 illustrates the idea. By identifying the linear offset between two patterns in polar coordinates, we can determine the angle by which the patterns differ in the $x-y$ plane. Because a polar image is still a bitmap image, we again make use of template matching techniques to determine the offset between two polar images. After the polar analysis, the patterns are aligned properly in the $x-y$ plane by a single rotation, and compared using the four template classifiers mentioned above.

Polar Transform

The polar coordinates of a point in the x-y plane are given by the point's radial distance, r, from the origin and the angle, θ, between that radius and the x axis. The well known relations are:

$r = \sqrt{(x-x_o)^2 + (y-y_o)^2}$
and
$\theta = tan^{-1}(\frac{y-y_o}{x-x_o})$, where (x_o, y_o) is the origin.

A symbol originally drawn in the screen coordinates (x-y plane) is transformed into polar coordinates by applying these formulae to each of the points. Figure 3a illustrates a typical transformation. As shown in Figure 3b, when a pattern is rotated in the x-y plane, the corresponding polar

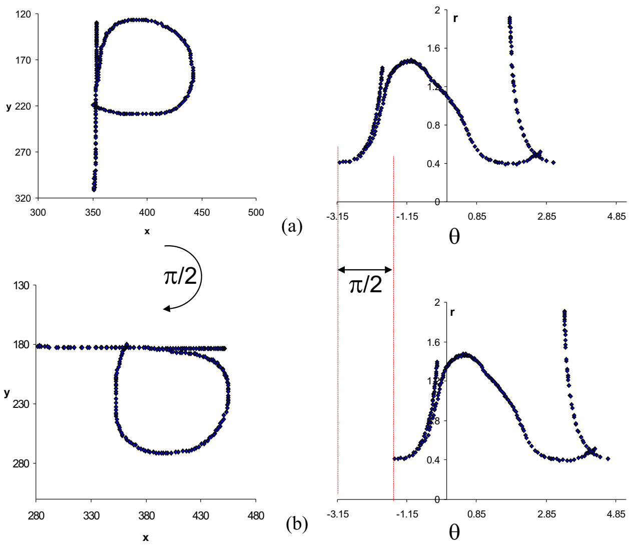


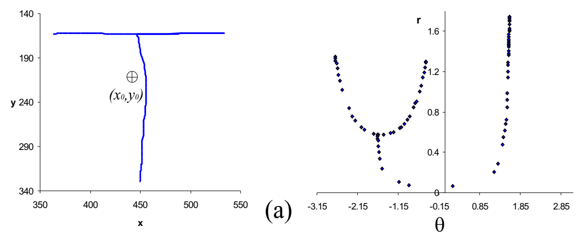
Figure 3: (a) Left: Letter 'P' in screen coordinates. Right: in polar coordinates. (b) When the letter is rotated in the x-y plane, the corresponding polar transform shifts parallel to the θ axis.

image slides parallel to the θ axis by the same angular displacement.

To find the angular offset between two polar images, we use a slide-and-compare algorithm in which one image is incrementally displaced along the θ axis. At each displacement, the two images are compared to determine how well they match. The displacement that results in the best match indicates how much rotation is needed to best align the original images. Because the polar images are in fact 2D binary patterns (48x48 quantized templates), we can use the template matching techniques described earlier to match the polar images. In particular, we use the modified Hausdorff distance (MHD) as it is slightly more efficient than the regular Hausdorff distance (directed distances need not be sorted), and it performs slightly better than the Tanimoto and Yule coefficients in polar coordinates.

One difficulty with the polar transform is that data near the centroid of the original image is sensitive to the precise location of the centroid. Consider Figure 4 which shows two similar shapes and their polar transforms. In the top image the tail of the "T" curves slightly to the left while in the bottom image it curves slightly to the right. This difference causes the image centroids to fall on the opposite sides of the tail, which, in turn leads to significant dissimilarity in the polar transforms for small r values. Naturally, the modified Hausdorff distance is adversely affected by these variations.

To alleviate this problem, we introduce a weighting function $w(\cdot)$ that attenuates the influence of pixels near the centroid of the screen image. Using this function, the directed MHD becomes:



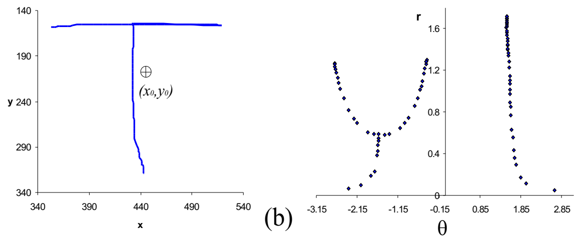


Figure 4: For small values of r, the θ coordinate is sensitive to the precise location of the centroid of the screen image. (a) Letter 'T' and its polar transform. (b) Nearly the same letter except for the curl of the tail. The difference in curl causes noticeable differences in the polar transforms for small values of r.

$$h_{mod\_weighted}(A,B) = \frac{1}{N_a}\sum_{a\in A} w(a_r)\cdot \min_{b\in B} \|a-b\|$$

where a_r represents the radial coordinate of point a in the quantized polar image A. The directed distance from B to A, $h_{mod\_weighted}(B,A)$, is calculated similarly. The maximum of the two directed distances is the MHD between A and B. Our weighting function has the form:

$$w(r) = r^{0.10}$$

The exponent in the function has been determined experimentally for best performance. The function asymptotes at 1 for large values of r, and falls off rapidly for small values of r. By assigning smaller weights to the pixels near the centroid, this function allows the Hausdorff distance to be governed by the pixels that reside farther from the centroid, hence reducing the sensitivity to the precise centroid location.

Polar Transform as a Pre-Recognizer

The degree of match between two polar images provides a reasonable estimate of the match of the original screen images. In fact, if it were not for the imprecision of the polar transform for small r values, the entire recognition process could be performed exclusively in the polar plane. The match in polar coordinates discounts data near the centroid of the screen image, which can result in false positive matches (*i.e.*, declaring a close match between two patterns when they are in fact dissimilar), but it rarely results in false negative matches. Thus, the polar analysis can be used as a pre-recognition step to eliminate unlikely definitions. In practice, we have found that the correct definition for an unknown is among the definitions ranked in the top 10% by the polar coordinate matching. Thus, we discard 90% of the definitions before considering the match in screen coordinates.

This approach is conceptually similar to cascading presented in (Alimoglu & Alpaydin 2001), where a simple classifier is used to reduce the number of classes before a more complex classifier with a more expensive classification rule is applied. In our case, however, the polar transform not only serves as a pre-elimination step but also as a means to efficiently achieving rotation invariance. We have found this dual functionality of the polar transform to be invaluable for achieving real-time performance on an otherwise computationally demanding task.

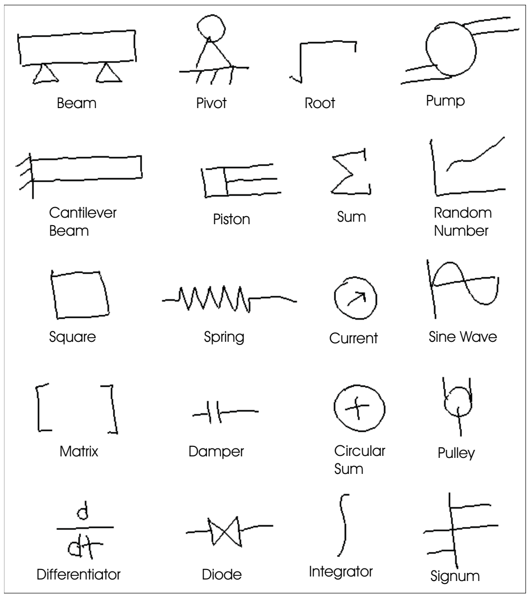


Figure 5: Symbols used in the graphic symbol recognition experiment.

User Studies

	Top 1 (%)	Top 2 (%)	Recog. Time (ms)
Test1	90.7	96.3	332
Test2	95.7	98.3	354
Test3	94.7	97.3	623
Test4	98.0	99.0	674

Table 1: Results from the graphic symbol recognition study. All tests were conducted on a 2.0 GHz Pentium 4 machine with 256 MB of RAM.

We developed a computer program that implements the approach described above and conducted a user study to assess its performance. Users were asked to draw the 20 symbols shown in Figure 5. The study included five users, who each provided three sets of these symbols. Because the participants had little or no experience using the digitizing tablet and stylus, they were allowed to acquaint themselves with the hardware until they felt comfortable, which typically took about 2 to 3 minutes. Each experimental session involved only data collection; the data was processed at a later time. This approach was chosen to prevent participants from adjusting their writing style based on our program's output.

Four different types of tests were conducted using the collected data. The tests differ based on (1) the number of definition symbols used for training, and (2) whether the test was conducted in a user-dependent or user independent manner. Below we detail each of these tests and the results.

Test 1: Single definition set, user-dependent: In this test, the recognizer was evaluated separately for each user. Each test consisted of three iterations, akin to the K-fold cross validation technique with K=3. In each iteration, one of the user's three sets of symbols was used for training, and the other two were used for testing. Different iterations employed different test sets. The performance for each user was computed as the average of the three iterations. The first row of Table 1 shows the results obtained from this study, averaged over the five users. In this table, the first column shows the recognition accuracy, or the rate at which the class ranked highest by the recognizer, is indeed the correct class. We call this the "top-one" accuracy. The second column shows the "top-two" accuracy, or the rate at which the correct class is either the highest or second highest ranked class. The last column shows the average recognition time in milliseconds.

Test 2: Two definition sets, user-dependent: This test is similar to the first test except, in each of the three runs, *two* sets of symbols were used for training while the remaining set was used for testing. Hence, during recognition, each unknown was compared to 40 definition symbols – 2 definitions per symbol. As shown in the second row of Table 1, the additional training set increased the recognition accuracy at the expense of only a minor increase in the recognition times.

Test 3: Twelve definition sets, user-independent: The aim of this test was to evaluate the recognizer when the training and test sets belonged to different users. When testing a particular user's data, the training database consisted of all users' symbol sets excluding the data from the user under consideration. In each run, the database thus consisted of a total of twelve sets: three sets from each of the four users not involved in that particular test. This test mimics a walk-up-and-draw scenario in which the user works directly from a pretrained recognizer without providing his or her own training symbols. The third row of Table 1, shows the performance obtained in this setting.

Test 4: Fourteen definition sets, partially user-dependent: The difference between this test and the previous one is that the training database contained two symbol sets from the user under consideration, in addition to the twelve sets from other users. In terms of training sets employed, this experiment is thus a hybrid of Test 2 and Test 3. As shown in the last row of Table 1, the top-one accuracy in this case reaches 98%.

We believe that the results of our user study are quite promising when compared to results reported in the literature. For example, Landay & Myers (2001) report a recognition rate of 89% on a set of 5 single-stroke editing gestures. In our case, however, there are 20 symbol definitions which can be drawn with any number of strokes. In a different study involving 7 multi-stroke and 5 single-stroke shapes, Fonseca & Jorge (2000) report recognition rates around 92% in an experiment where half of the subjects were experts in using the hardware. On a database of 13 symbols, Hse & Newton (2003) report a recognition rate of 97.3% in a user-dependent setting and 96.2% in a user-independent setting, where each symbol was trained using 30 examples. In a user-dependent setting, we achieve an accuracy of 95.7% on a database of 20 symbols where each symbol was trained with 2 examples (Test-2), and 94.7% in a user-independent setting where each symbol was trained with 12 examples (Test-3).

To evaluate the efficiency of our polar coordinate analysis, we conducted a separate experiment in which the angular alignment of the images was computed in screen coordinates via incremental rotations. This not only bypassed the polar coordinate approach for computing optimal alignment, but also bypassed the accompanying pre-recognition step in which unlikely definitions are pruned. With these modifications, the average recognition time for Test 1 increased to 3590ms, while the recognition accuracy remained the same. This suggests that the polar analysis provides significant savings in overall processing time without any decrease in accuracy.

Related Work

The problem of hand-drawn symbol recognition has recently attracted many other researchers. Closely related to our work is that of (Veselova & Davis 2004), (Hammond & Davis 2003), (Hse & Newton 2003), (Fonseca, Pimentel, & Jorge 2002), (Calhoun *et al.* 2002), (Matsakis 1999), (Gross 1994), (Apte, Vo, & Kimura 1993) and (Rubine 1991). Some of these systems require the shape descriptors to be manually encoded while others require a large set of training examples to reliably learn new definitions. Other limitations include restrictions to single-stroke symbols, rotation dependence, or reliance on a segmentation process. Our techniques were designed to alleviate many of these issues.

Concluding Remarks

We have described a trainable, multi-stroke, hand-drawn symbol recognizer designed to be used in sketch-based interfaces. With our techniques, symbol definitions can be learned from single prototype examples, allowing users to train new symbols or adjust existing ones on the fly. Our approach avoids a number of problematic issues in symbol recognition, such as segmentation and feature extraction. Also, our approach is tolerant of overtracing, missing and extra pen strokes, variations in line style, and variations in drawing order.

Our recognizer employs a two-step recognition scheme. First, polar coordinates are used to efficiently determine angular alignment and eliminate unlikely definitions. Then, the remaining definitions are examined in screen coordinates using an ensemble of four template classifiers. Each classifier produces a list of definitions ranked according to their similarity or dissimilarity to the unknown symbol. The results of the individual classifiers are combined to produce the recognizer's final decision. Our experiments have demonstrated that this two-step approach is an order of magnitude more efficient than performing the alignment and recognition solely in screen coordinates.

We conducted a user study of our recognizer and found that it can accurately classify symbols with only a small amount of training data. Furthermore, our recognizer worked well in both user-dependent and user independent settings.

References

Alimoglu, F., and Alpaydin, E. 2001. Combining multiple representations for pen-based handwritten digit recognition. *ELEKTRIK: Turkish Journal of Electrical Engineering and Computer Sciences* 9(1):1–12.

Apte, A.; Vo, V.; and Kimura, T. D. 1993. Recognizing multistroke geometric shapes: An experimental evaluation. In *UIST 93*, 121–128.

Calhoun, C.; Stahovich, T. F.; Kurtoglu, T.; and Kara, L. B. 2002. Recognizing multi-stroke symbols. In *AAAI Spring Symposium on Sketch Understanding*, 15–23.

Cheung, K.-W.; Yeung, D.-Y.; and Chin, R. T. 2002. Bidirectional deformable matching with application to handwritten character extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(8):1133–1139.

Dubuisson, M.-P., and Jain, A. K. 1994. A modified hausdorff distance for object matching. In *12th International Conference on Pattern Recognition*, 566–568.

Fligner, M.; Verducci, J.; Bjoraker, J.; and Blower, P. 2001. A new association coefficient for molecular dissimilarity. In *The Second Joint Sheffield Conference on Chemoinformatics*.

Flower, D. R. 1998. On the properties of bit string-based measures of chemical similarity. *Journal of Chemical Information and Computer Science* 38:379–386.

Fonseca, M. J., and Jorge, J. A. 2000. Using fuzzy logic to recognize geometric shapes interactively. In *Proceedings of the 9th Int. Conference on Fuzzy Systems (FUZZ-IEEE 2000)*.

Fonseca, M. J.; Pimentel, C.; and Jorge, J. A. 2002. Cali-an online scribble recognizer for calligraphic interfaces. In *AAAI Spring Symposium on Sketch Understanding*, 51–58.

Gross, M. D. 1994. Recognizing and interpreting diagrams in design. In *ACM Conference on Advanced Visual Interfaces.*, 88–94.

Hammond, T., and Davis, R. 2003. Ladder: A language to describe drawing, display, and editing in sketch recognition. In *2003 International Joint Conference on Artificial Intelligence (IJCAI)*.

Hse, H., and Newton, A. R. 2003. Sketched symbol recognition using zernike moments. Technical report, EECS, University of California.

Kimura, T. D.; Apte, A.; and Sengupta, S. 1994. A graphic diagram editor for pen computers. *Software Concepts and Tools* 82–95.

Kittler, J.; Hatef, M.; Duin, R. P. W.; and Matas, J. 1998. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(3):226–239.

Landay, J. A., and Myers, B. A. 2001. Sketching interfaces: Toward more human interface design. *IEEE Computer* 34(3):56–64.

Matsakis, N. E. 1999. *Recognition of Handwritten Mathematical Expressions*. Master thesis, MIT.

Miller, E. G.; Matsakis, N. E.; and Viola, P. A. 2000. Learning from one example through shared densities of transforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 464–471.

Ozer, O. F.; Ozun, O.; Tuzel, C. O.; Atalay, V.; and Cetin, A. E. 2001. Vision-based single-stroke character recognition for wearable computing. *IEEE Intelligent Systems and Applications* 16(3):33–37.

Rubine, D. 1991. Specifying gestures by example. *Computer Graphics* 25:329–337.

Rucklidge, W. J. 1996. *Efficient Visual Recognition Using the Hausdorff Distance*. Number 1173 Lecture Notes in computer Science,. Berlin: Springer-Verlag.

Sim, D.-G.; Kwon, O.-K.; and Park, R.-H. 1999. Object matching algorithms using robust hausdorff distance measures. *IEEE Transactions on Image Processing* 8(3):425–429.

Tubbs, J. D. 1989. A note on binary template matching. *Pattern Recognition* 22(4):359–365.

Veselova, O., and Davis, R. 2004. Perceptually based learning of shape descriptions for sketch recognition. In *The Nineteenth National Conference on Artifical Intelligence (AAAI-04)*.

Yasuda, H.; Takahashi, K.; and Matsumoto, T. 2000. A discrete hmm for online handwriting recognition. *International Journal of Pattern Recognition and Articial Intelligence* 14(5):675–688.