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.

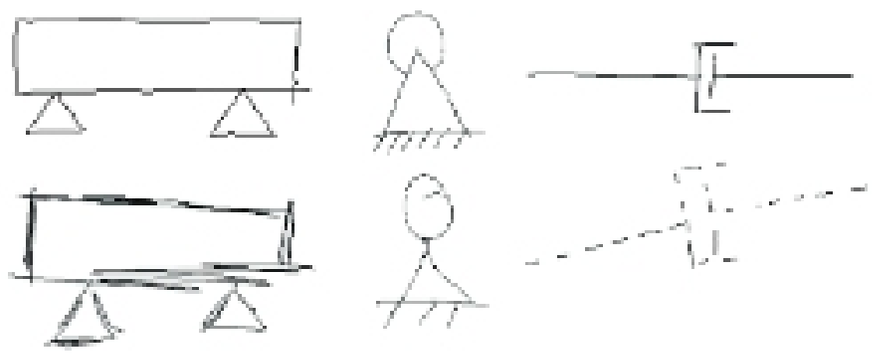

Figure 1: Examples of symbols correctly recognized by our system. The top row shows symbols used in training and the bottom row shows correctly recognized test symbols. At the time of the test, the database contained 104 definition symbols.

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).

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.

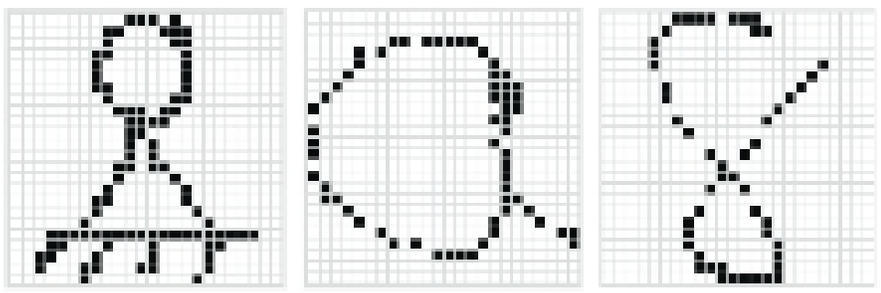

Figure 2: Examples of symbol templates: A mechanical pivot, letter 'a', digit '8'. The examples are demonstrated on 24x24 templates to better illustrate the quantization.

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

$$h(A,B) = \max_{a \subset A}(\min_{b \subset B} \|a-b\|)$$

$\|a-b\|$ 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:

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

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:

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

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 $1/15^{th}$ 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:

$$minscore^r = \min_{k=1}^{M} {d_k}^r \ , \ maxscore^r = \max_{k=1}^{M} {d_k}^r$$

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

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