

Discriminating Stroke Features Experiments

Introduction

This document describes the progress made to date on the discriminating stroke features design. The approach for this filter in the system was to determine which of the strokes are most discriminating and then use those strokes to generate new words. The idea was that if a letter is composed of strokes, then by knowing which strokes are most discriminating, we will also know which letters are therefore most discriminating. From this, we can generate new words at random, composed of these letters. Presumably, it would be more difficult to forge a word generated at random than a signature because knowledge of the whole writing style for a particular writer would be needed to be able to forge new words on the fly.

Approaches

Approach 1 – Stroke to Letter Alignment

The original design approach was to segment the strokes from the word, and associate strokes to letters. The next step would perform analysis on the letters to see which ones are statistically more discriminating. The difficulty in this approach is, however, in dividing the words into letters consistently.

The first attempt was to evenly divide the word, based on the number of known letters in the word. The letters are known to the system, because at enrollment time, they are stored with the word that is to be written. If the majority of the points in a stroke were close to this line, then the stroke belonged to the letter. However, this approach proved to be inaccurate, as letters were not always evenly spaced in written words.

Next, I attempted to use clustering to find the centers of the points in each letter. The number of clusters was set to the number of letters in the word, and the centers of the clusters were set to the evenly divisions as before. Then, K-means clustering was run, considering each point to determine the cluster centers for each letter. These became the new centers of the letters in the word. An example of this is shown in Figure 1, the darker lines represent the cluster-aligned letter centers and the lighter colored lines represent the evenly spaced alignments.

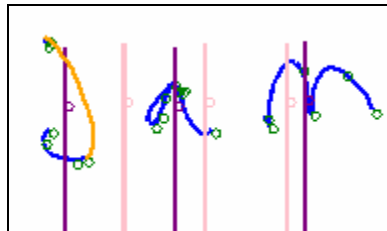


Figure 1 - Letter to Stroke Matching

This approach however, has flaws in that it does not always match strokes correctly to letters. Points in each stroke are assigned to a cluster based on their distance from that cluster and the majority voted cluster is the letter assigned. In Figure 2 below, the yellow highlighted stroke, part of the letter, 'P', is erroneously assigned to the letter 'a' cluster. Similarly, in Figure 3, the first stroke in the letter, 'w' is assigned to the previous letter, 'e'. As such, the use of clustering only

slightly improved the accuracy of stroke to letter assignments, but not by enough to make a difference.

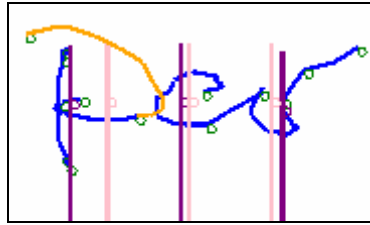


Figure 2 - Letter to Stroke Mismatch

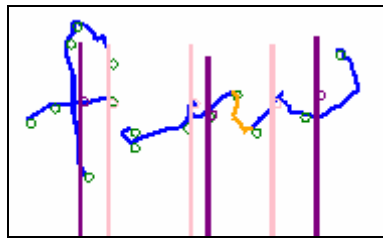


Figure 3 - Letter to Stroke Mismatch

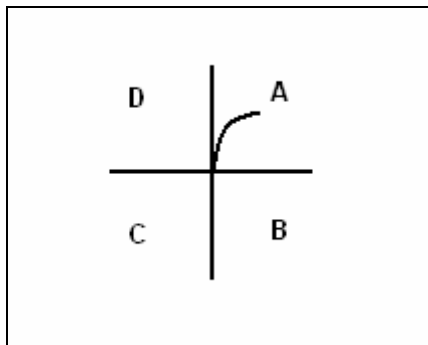


Figure 4 - Stroke Encoding

Each stroke was encoded in the same manner as in the second filter, such that a letter composed of one or more strokes could be represented by a string. To encode a stroke, the start point was placed at the origin in the x, y, plane, with the end-point landing in one of four quadrants – see Figure 4.

Central to this approach was being able to compare encoding of letters within a given writer's samples and across writers to determine which letters are most specific to a given writer. However, after reviewing the segmentation results above, and the stroke encoding mappings to each letter, there is not enough consistency to statistically make any accurate comparison. For instance, in Figure 5 below, we can see a few letters for a single signer, and the resulting encodings using the process above. Note that very few of the encodings match for the 3 letter samples. In fact, much of the data is this way.

Ten words were gathered for each signer, composing one or a few samples of each of the 22 letters in the alphabet. Given much more data per signer, this approach might have worked better. Even better would be a more accurate way of dividing strokes and letters per signer.

SignerID	Letter	Encoding
1	a	CDBC
1	a	ABCACA
1	a	ACAA
1	a	AACA
1	a	C
1	a	ABDCABA
1	a	AACA
1	a	BB
1	b	ACAC
1	b	A
1	c	AB
1	c	ACA
1	c	CBC
1	c	ACAC

Figure 5 - Letter Stroke Encodings

Approach 2 – Stroke To Stroke Comparison

Another approach considered, but not yet attempted, is to compare raw strokes to raw strokes, rather than try to match strokes to letters and compare letters. The idea is that if a writer writes a particular type of stroke consistently (for instance upward strokes), then why not just compare strokes in one word to the same strokes in another word? If these strokes matched below some empirically determined threshold then the writer passes as valid.

Granted, this approach will likely require a more complicated encoding mechanism to capture more than just the stroke direction, but also some notion of length, or perhaps velocity. The idea is to build a database of template or training strokes from the set of all enrollment or training words. In this case the word and the letter that a stroke came from is not relevant, only the raw stroke itself. Then, as new words come in for testing, segment them into strokes as well. For each stroke in the new word, compare it against each template stroke of that type in the reference set. Pick the best match. If the sum of best matches for all strokes in the test word is below some defined per letter threshold, then the writer should pass.

The comparison could be a dynamic time warping comparison of every point in both strokes, or it could be feature based, comparing features such as velocity in each stroke, length and average pressure in the stroke. This would likely be more discriminating than simply comparing stroke direction. Perhaps a combination of the two mechanisms could be combined to create a notion of distance between two strokes. The high level algorithm is as follows:

```
Given a Training Set, R:
    For every word in the training set
        pull out all strokes and stroke features.
        store strokes by encoding, per writer

Given a Test Word, W:
    For every stroke, Si, in W:
        Compare against all templates in R of the same type as Si
        Pick the best match
    Sum the total of matches
    If normalized total < writer threshold, word passes!
```

Discriminating Strokes

A key idea in this filter's design is to not only compare strokes, but to be able to determine in advance the strokes that are most discriminating for a particular writer. One approach considered is the use of a probabilistic method. Of course, this implies that a notion be defined for strokes which makes them consistent statistically. The approach is as follows: find the strokes that one writer writes consistently, and it is fine if a few other writers in the reference set also write this way. Then, from that set of writers, find a stroke that *only the test writer* writes consistently. This would serve to differentiate that writer from all others. Find one or more strokes in this way. Then, generate new test words dynamically from combinations of these strokes for a new writer to attempt. Following is a high-level view of that algorithm, defined recursively:

```
FindUniqueStrokeSet( Writer Wt, ReferenceSet R)
BEGIN

For Each Stroke, Si in R
    // Get the probability that in the ReferenceSet, Writer Wt wrote Si uniquely
    OtherWriters = All Writers – Wt
    Prob = FindProbUniqueStroke(OtherWriters, Si, R)
    MaxProb = Max( MaxProb, Prob)
    StrokeList.Add( MaxProb)
End For

Sort StrokeList by Prob decreasing
Pick top n stroke probabilities
Form New Word from Strokes
END

FindProbUniqueStroke(Set of Writers W, Stroke S, ReferenceSet R)
BEGIN
// Count up how many times this set of writers writes this stroke in this manner
// Use some notion of distance to determine if they write it that way
CountSi = FindProbOfStroke( Si, W)
ProbSi = 1 / (CountSi) + 1

if( depth > 4 or ProbSi == 1)
    return ProbSi, StrokeList
else
    depth ++
    // Find the prob of the next stroke also
    return ProbSi * FindProbUniqueStroke( W, S+1, R)
END
```

Conclusions

A number of approaches were investigated for determining discriminating strokes and generating new words dynamically. Overall, I believe the idea was sound, but more work needs to be spent developing the best way to determine the strokes that compose letters (or consider all strokes in the word, ignoring letters) and the notion of similarity between strokes. Given more experimentation, it is likely that these techniques could be refined.

References

- [1] WACOM Intuos Pen with tilt feature:
<http://wacomdirect.wacom.com/wacomdirect/product.asp?dept%5Fid=100&sku=XD45USB>
- [2] Logitech IO Pen:
<http://www.logitech.com>
- [3] Schomaker, L. & Segers, E. (1999). *Finding features used in the human reading of cursive handwriting* International Journal on Document Analysis and Recognition, 2, 13-18.
- [4] Schomaker, Lambert (April 1, 1996). Pen Tip Based velocity in handwriting. Set of tools:
<http://hwr.nici.kun.nl/unipen/hwr-tutor/velocity.html>
- [5] Kai Huang, Hong Yan. Stability and style variation modeling for on-line signature verification. Pattern Recognition 36 (2003) 2253-2270.
- [6] Anil K. Jain, Freiderike D. Griess, Scott D. Connell. On-line signature verification Pattern Recognition 35 (2002) 2963-2972.
- [7] Jaeyeon Lee, Ho-Sub Yoon, Jung Soh, Byung Tae Chun, Yun Koo Chung. Using geometric extrema for segment-to-segment characteristics comparison in online signature verification. Pattern Recognition 37 (2004) 93-103.
- [8] Alan McCabe. Hidden Markov Modelling with Simple Directional Features for Effective and Efficient Handwriting Verification.
- [9] Gobal Gupta, Alan McCabe. A review of dynamic handwritten signature verification. September 1997.
- [10] Vishvjit S. Nalwa. Automatic On-line Signature Verification. Proceedings of the IEEE, Vol. 85, No. 2, February 1997.
- [11] Rejean Plamondon and Guy Lorette. Automatic Signature Verification and Writer Identification - The State of the Art. Pattern Recognition #22, 1989. pp. 107-131
- [12] H.E.S. Said, T.N. Tan, K.D. Baker. Personal identification based on handwriting. Pattern Recognition 33 (2000) 149-160.

[13] Charles C. Tappert, Ching Y. Suen, Toru Wakahara. The State of the Art in On-Line Handwriting Recognition. IEEE Transactions on pattern analysis and machine intelligence. Vol 12, No. 89 August 1990.