# Algorithms for matching hand-drawn sketches

D. Lopresti, A. Tomkins and J. Zhou

Panasonic Technologies, Inc., 2 Research Way, Princeton, NJ 08540, USA

## 1. INTRODUCTION

Searching a database for close matches to a query is a fundamental problem in pattern recognition and information retrieval. In two earlier papers [3, 2], Lopresti and Tomkins describe algorithms for matching *pictograms*, small labels of electronic ink used to give files graphical names in pen-based systems. This idea was later extended to searching lengthy handwritten notes for keywords ("word spotting") [4]. The basic approach in this case is to segment the input into strokes and extract vectors of descriptive features (*e.g.*, stroke length, total angle traversed). These feature vectors are then mapped into a small set of basic types using vector quantization (VQ). The effect is that handwriting is represented as a string over an alphabet of stroke types. Approximate string matching techniques are then used to perform comparisons between queries and pre-determined databases.

This notion of treating electronic ink as a temporal sequence of pen strokes – without attempting to "recognize" it – offers several advantages, including language-independence. Algorithms for matching ink are more closely related to on-line signature verification [7] than they are to traditional two-dimensional object recognition.

Other researchers have begun examining similar ideas. Hull, Reynolds, and Gupta describe a system that combines the outputs from three different matching algorithms with the goal of producing more accurate results [1]. Poon, Weber, and Cass present a Macintosh-based application called "Scribbler" that allows a user to search for words and simple graphical symbols [8].

These approaches have been shown to work well for ink that is primarily textual. However, when processing more complicated pictorial data, certain sub-structures within a larger image will correspond stroke-for-stroke, but these basic "motifs" may be drawn by the user in an otherwise arbitrary order. Figure 1 demonstrates this: in the top sketch on the left side, the house is drawn first, then the tree, then the car, whereas in the very similar sketch on the right side, the drawing order is reversed. Moreover, if the goal is to search a database, the best match may be "partial" in the sense that certain elements are omitted or repeated (see, for example, Figure 2). Algorithms that have been developed for matching textual ink are not flexible enough to capture these kinds of *block motion*.

In a recent paper, Lopresti and Tomkins introduced a family of algorithms for block string matching that address this issue [5]. Preliminary results, based on a small experiment, seemed promising [6]. In the present paper, we give a more detailed analysis of the problem of matching hand-drawn sketches. We describe a hierarchical approach that exploits the temporal nature of electronic ink, and evaluate it using a larger dataset.

Figure 1 presents a conceptual overview. Sketch matching can be viewed at several levels of abstraction. At the highest level, large subsequences of the ink string ("blocks" or "motifs") may be interchanged and repeated to allow for variations in drawing order. At the next level, a single motif is treated as a string of strokes that are always drawn in a consistent order. Finally, at the lowest level, individual pen strokes are matched using one of a number of possible techniques.

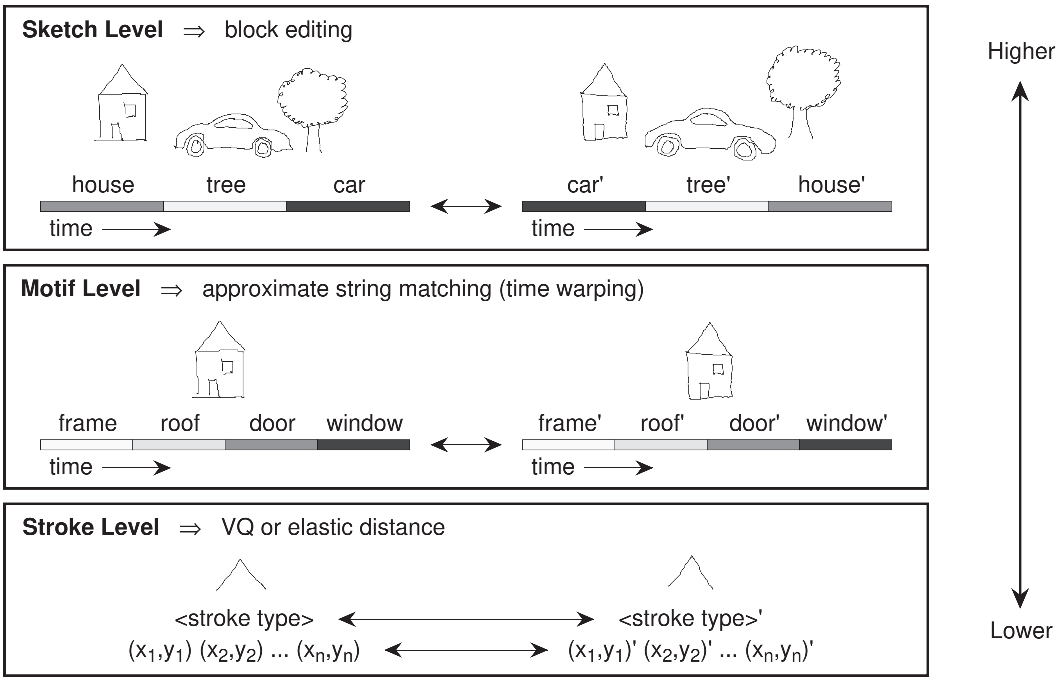


Figure 1. Levels for sketch matching.

## 2. SKETCH LEVEL MATCHING

At the highest level, the algorithm partitions the ink string corresponding to the query into blocks dynamically, and matches each block to a database block such that the sum of the distances between corresponding blocks is minimized. Thus, the procedure allows high-level block motion by placing arbitrary blocks in correspondence, and incorporates low-level string edits in an underlying distance function.

Let $Q = q_1 q_2 \ldots q_m$ and $D = d_1 d_2 \ldots d_n$ be the query and database ink strings, respectively, over an alphabet of pen strokes, $q_i, d_j \in \Sigma$. A *t-block substring family of* $Q$, $Q|_t$, is a multiset containing $t$ substrings of $Q$. The *block edit distance* $\mathcal{B}$ between two strings $Q$ and $D$ is determined by finding the best way to choose substring families of $Q$ and $D$ and correspond each member of $Q|_t$ with some member of $D|_t$. For each pairing, a cost is assessed based on the distance between the two substrings. The correspondence between blocks is given by a permutation $\sigma \in S_t$ from the symmetric group on $t$ elements. More formally,

$$\mathcal{B}(Q, D) \equiv \min_{t} \min_{Q|_t, D|_t} \min_{\sigma \in S(t)} \left\{ \sum_{i=1}^{t} dist\left(Q^{(i)}, D^{(\sigma(i))}\right) \right\} \tag{1}$$

While the most general form of Equation 1 is NP-complete [5], certain variants have efficient solutions. In particular, if the substring family for one of the ink strings, say $D$, is unconstrained so that: (1) blocks may overlap, and (2) all of $D$ need not be used, we have developed the following polynomial time algorithm. Let $W(i, j)$ be the value of the best possible match between the substring $q_i \ldots q_j$ and any substring of $D$:

$$W(i, j) \equiv \min_{k \leq l} \{dist(q_i \ldots q_j, d_k \ldots d_l)\} \tag{2}$$

Now define $\mathcal{M}(i)$ to be the best block match between $q_1 \ldots q_i$ and $D$. We can then optimally pair blocks of the query string with blocks of the database string using the following recurrence:

$$\mathcal{M}(i) = \min_{j<i} \{\mathcal{M}(j) + W(j+1, i)\} \tag{3}$$

Here $W(j+1, i)$ allows the best match in the database string corresponding to $q_{j+1} \ldots q_i$ to be added to the optimal solution for $q_1 \ldots q_j$ (*i.e.*, $\mathcal{M}(j)$), for "cuts" at all possible indices $j$ in the query string. Once we have computed $\mathcal{M}$ for $i = 1, 2, \ldots, m$, our final answer is $\mathcal{B}(Q, D) = \mathcal{M}(m)$.

## 3. MOTIF LEVEL MATCHING

Here we assume that an algorithm at the sketch level has chosen candidate intervals from the query and database strings, and we must compare these two substrings. There will be no high-level rearrangements necessary, but there might be substantial local editing required to find the similarity between the two substrings. We break the point sequences into strokes using a segmentation algorithm whose granularity will depend on the stroke-level distance measure. We then use dynamic programming to determine the edit distance between the induced sequences of strokes. The cost of a deletion or insertion is a function of the "size" of the ink in question. The cost of a substitution is given by the stroke-level distance measure.

When the underlying stroke distance measure allows it, we augment traditional string edit distance with two new editing operations: splitting one stroke into two, and merging two strokes into one. These account for imperfections in the segmentation.

More formally, let $Q = q_1 q_2 \ldots q_m$ and $D = d_1 d_2 \ldots d_n$ be intervals from the query and database ink strings, respectively. Let $dist_{i,j}$ represent the cost of the best match between the first $i$ strokes of $Q$ and the first $j$ strokes of $D$. The recurrence is then:

$$dist_{i,j} = \min \begin{cases} dist_{i-1,j} & + \ c_{del}(q_i) \\ dist_{i,j-1} & + \ c_{ins}(d_j) \\ dist_{i-1,j-1} & + \ c_{sub}(q_i, d_j) \\ dist_{i-1,j-2} & + \ c_{split}(q_i, d_{j-1} d_j) \\ dist_{i-2,j-1} & + \ c_{merge}(q_{i-1} q_i, d_j) \end{cases} \qquad 1 \leq i \leq m, \ 1 \leq j \leq n \tag{4}$$

## 4. STROKE LEVEL MATCHING

At this level, we must compare a query stroke and a database stroke extracted by a higher-level algorithm. The stroke-level distance function determines whether the points contained in each stroke represent similar pen-tip trajectories. While there might be slight differences in the pen motion, no significant rearrangements will be necessary to reveal the similarities.

We have studied two potential measures for stroke-level matching: VQ distance and elastic match distance. The VQ distance between a query stroke $Q$ and a database stroke $D$ is computed as follows. First, a vector of 13 real-valued features is derived from each stroke. Next, each vector is compared to a set of 64 vectors representing canonical strokes, and is classified as the most similar type. The final distance is then defined to be the distance between the two stroke types, as given by a weighted Euclidean distance on the feature space. This is the same metric we used previously for searching textual ink [4].

In the case of elastic match distance, let $Q = q_1, q_2, \ldots, q_m$ and $D = d_1, d_2, \ldots, d_n$ be the two strokes in question, where each $q_i$ and $d_j$ is a point in the plane. The elastic distance between them is the edit distance between $Q$ and $D$ viewed as strings, where the substitution cost for $q_i$ and $d_j$ is simply the Euclidean distance: $c_{sub}(q_i, d_j) = \sqrt{(q_i.x - d_j.x)^2 + (q_i.y - d_j.y)^2}$.

## 5. EVALUATION

To test these techniques, we designed a simple experiment where purely local similarity measures would not be effective, but more powerful global approaches should perform well. Five subjects were asked to create an ink database consisting of 25 sketches, and another set of 25 ink queries to be applied against the database. For example, the instruction to create a database entry might be: "Draw a chair, a desk, a computer, and a lamp." The queries always consisted of either one or two motifs (*e.g.*, "Draw a computer and a chair."). An example of a query and its corresponding database entry is shown in Figure 2.

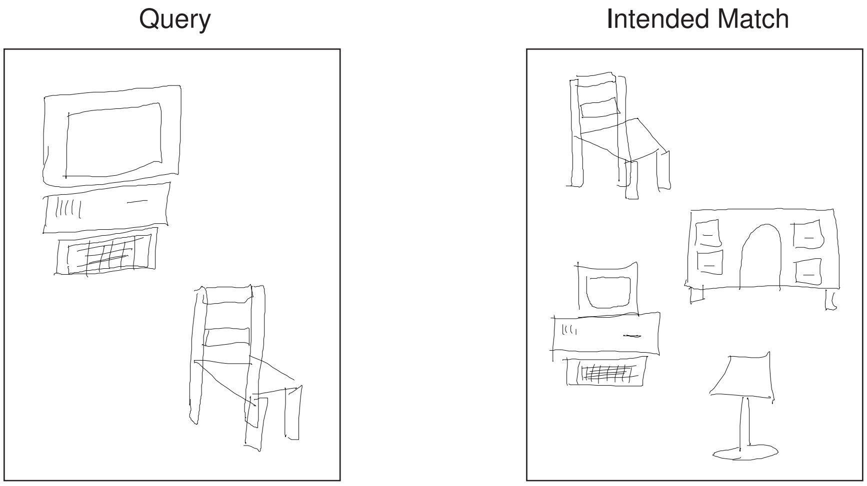


Figure 2. Example of a query and its intended match in the database.

Note that a given query might match multiple database entries. Of the 25 queries in our test, three had three matches in the database, nine had two matches, and 13 had only one match. Each database entry contained on-average 1,502 sampled points, while the queries averaged 713 points. A total of 279,210 ink points were collected for the experiment.

Figures 3 and 4 present the $3,125$ ($= 5 \times 25 \times 25$) distance values computed using our string block editing algorithm with underlying VQ and elastic match distances, respectively. The intended matches for each query are indicated by the "dots." While there is significant variability, both across queries and across subjects, it is fairly evident that elastic matching yields better results on-average (56% of the highest-ranked matches were true hits, versus only 30% for VQ distance). As we noted earlier, the VQ technique was designed for searching through handwritten text, and uses a representation based upon a space of "important" features. For the sketch matching problem, a measure such as elastic match distance that operates more firmly in the spatial domain seems better-suited.

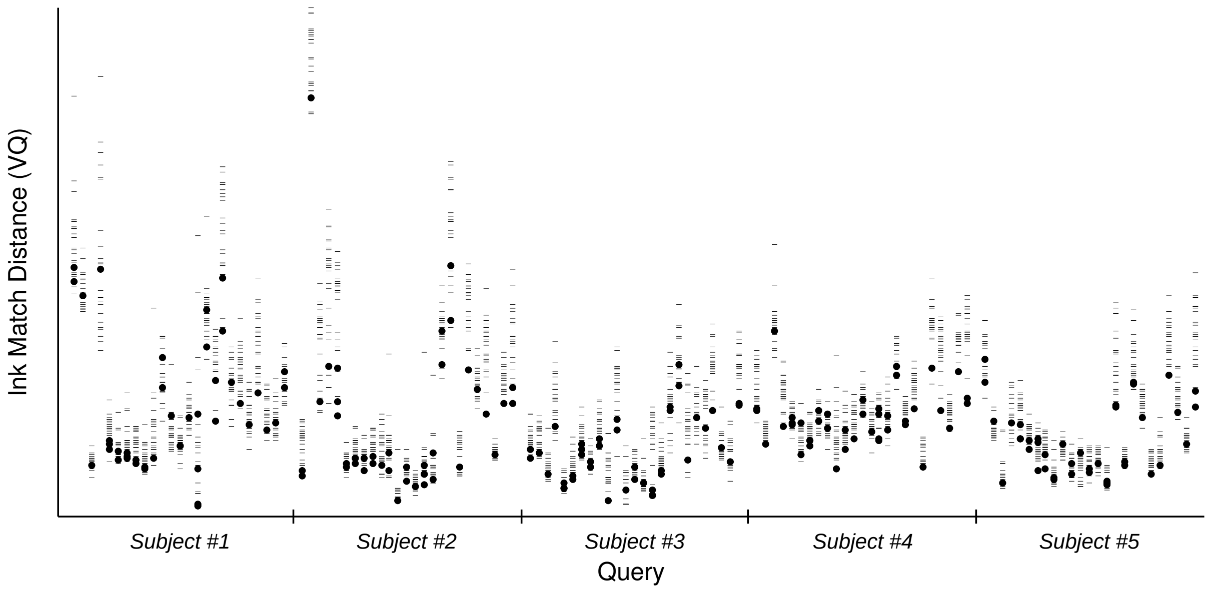


Figure 3. Results for sketch matching using underlying VQ distance.

## 6. CONCLUSIONS

Sketch matching in the temporal domain requires solving two difficult problems. First, reasonable choices for possible corresponding "motifs" must be found, and second, a good spatially-based matching algorithm must be used to compare them.

Our hierarchical approach addresses the former issue through the use of a string block editing model that is mathematically rigorous. At the lower levels, a number of possible techniques suggest themselves. To date we have examined two of these: VQ distance and elastic match distance.

Experimentally, the results we have obtained in our initial tests range widely. The algorithms sometimes perform impressively, despite the informal and highly variable nature

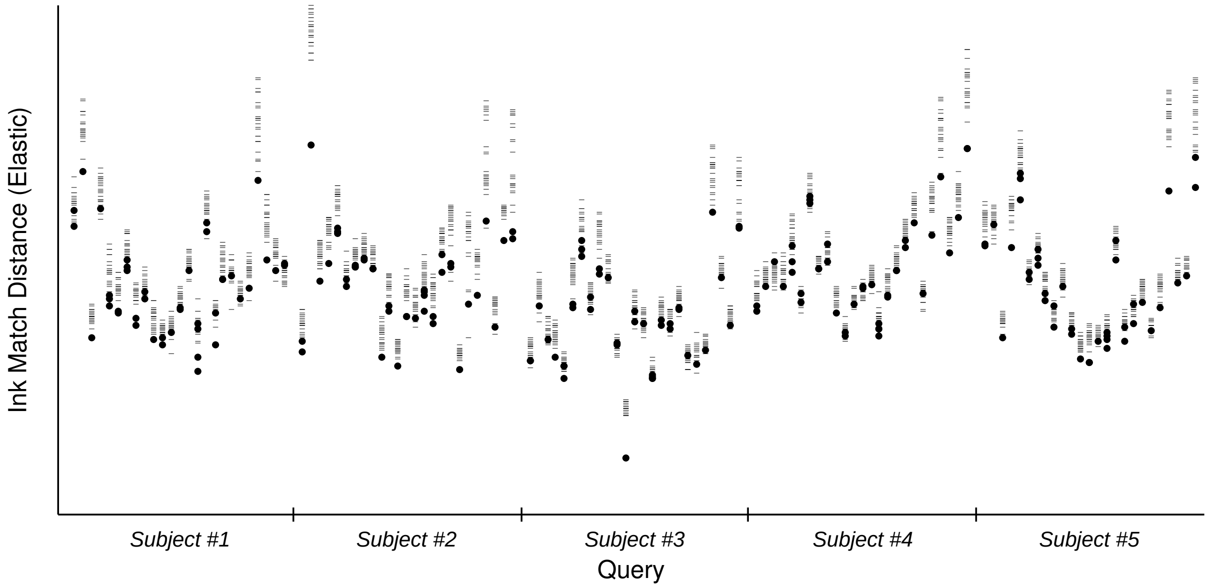


Figure 4. Results for sketch matching using underlying elastic match distance.

of the sketch data. On other occasions, the intended match is ranked much lower than obviously dissimilar database entries. We are currently studying ways of improving the quality and speed of our ink matching algorithms.

## 7. REFERENCES


[1] R. Hull, D. Reynolds, and D. Gupta. Scribble matching. In *Proc. Fourth Int. Work. Frontiers Handwriting Recognition*, pages 285–294, Dec. 1994.

[2] D. Lopresti and A. Tomkins. Approximate matching of hand-drawn pictograms. In *Proc. Third Int. Work. Frontiers Handwriting Recognition*, pages 102–111, May 1993.

[3] D. Lopresti and A. Tomkins. Pictographic naming. In *Adjunct Proc. Conf. Human Factors in Computing Systems*, pages 77–78, Apr. 1993.

[4] D. Lopresti and A. Tomkins. On the searchability of electronic ink. In *Proc. Fourth Int. Work. Frontiers Handwriting Recognition*, pages 156–165, Dec. 1994.

[5] D. Lopresti and A. Tomkins. Block edit models for approximate string matching. In *Proc. Second South American Work. String Processing*, pages 11–26, Apr. 1995. To appear in *Theoretical Computer Science*.

[6] D. Lopresti and A. Tomkins. Temporal domain matching of hand-drawn pictorial queries. In *Proc. Seventh Conf. Int. Graphonomics Society*, pages 98–99, Aug. 1995.

[7] R. Plamondon, editor. *Progress in Automatic Signature Verification*. World Scientific, Singapore, 1994.

[8] A. Poon, K. Weber, and T. Cass. Scribbler: A tool for searching digital ink. In *Human Factors in Computing Systems Conf. Companion*, pages 252–253, May 1995.