Ink as Multimedia Data

Daniel P. Lopresti
Bell Laboratories, Lucent Technologies Inc.
600 Mountain Avenue, Room 2C-552
Murray Hill, NJ 07974 USA

ABSTRACT

Treating electronic ink as first-class data – as opposed to simply a substitute for keyboard input – offers intriguing possibilities. The pen has well-known advantages in terms of portability and user acceptance, and ink is an extremely expressive medium that is inherently language-independent. A limitless range of representations are possible when writing/sketching. Moreover, ink shares important characteristics with other multimedia datatypes: it has a temporal component like speech, and a spatial component like images. However, adopting this approach raises some important issues with regard to storage and retrieval, as traditional techniques may no longer apply.

In this paper, we survey some of the research being done in this area. This includes work on representing, pre-processing, and compressing ink data. A particular challenge is the problem of retrieving previously saved ink documents, whether handwritten text or hand-drawn sketches. We review several algorithms adapted from approximate string matching that appear to be effective, and summarize the results of experiments reported in the literature. Systems that associate electronic ink, in the form of annotations, with other kinds of multimedia data will also be described. Finally, we conclude with a discussion of some open questions.

Keywords: electronic ink, handwriting recognition, pen computing, approximate ink matching, sketch matching, information retrieval.

1. INTRODUCTION

As an input device, the pen has numerous advantages over a keyboard. Nothing could be more natural than picking up a pen and jotting a quick note on a pad of paper. For languages such as Chinese which have thousands of characters in everyday use, keyboards are cumbersome devices accessible only to highly trained specialists. Even in the case of the Latin alphabet, the keyboard becomes a limiting factor as notebook computers continue to shrink in size. The pen also enjoys universal acceptance – while some people find a keyboard intimidating, a pen is a familiar, comfortable tool.

For the most part, today's pen computers operate in a mode that might be described as "eager recognition." Pen-strokes are translated as soon as they are entered, the user corrects the output of the recognizer, and then processing proceeds as if the characters had been typed on a keyboard. This has its advantages, chief among them that existing techniques can be used for storing and searching the data. This state of affairs is depicted in Figure 1.

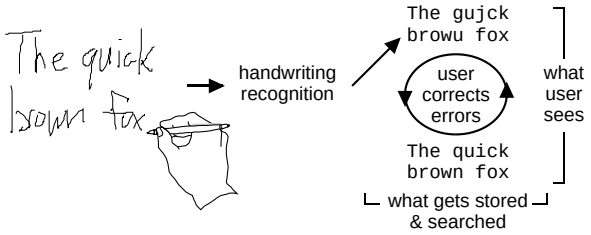

Figure 1: Traditional approach to pen computing.

The state-of-the-art in handwriting recognition has progressed significantly over the past several years [27,31]. Still, this problem has proved more difficult than first anticipated due to the large variation in the way people write. Human readers can make use of past experience as well as semantics when attempting to decipher written text; machines are not yet as adept. As a result, handwriting recognition is still an area of active research. Some of this work has focused on techniques to make it easier for the user to correct the errors that inevitably arise. Another recent approach is to make the problem simpler for the computer by changing the input alphabet so that confusing characters can be better disambiguated [9].

Even so, the necessity of having to proof-read continuously the output of a recognizer, pausing to correct whatever errors might occur, creates an overall frustrating experience at odds with the unconstrained way people are used to interacting with pen and paper. Nakagawa, et al.'s concept of "lazy recognition" evinces the negative impact this can have on creativity [26].

Instead of taking a very expressive medium, ink, and immediately mapping it into a small, pre-defined set of alphanumeric symbols, an alternative is to support ink as a first-class datatype [1]. There are compelling arguments for deferring or even eliminating handwriting recognition in certain applications:

1) Many of a user's day-to-day task can be handled entirely in the ink domain using techniques more accurate and less intrusive than handwriting recognition.
2) No existing character set captures the full range of graphical representations a human can create using a pen (e.g., sketches, maps, diagrams, equations, doodles). By not constraining pen strokes to represent "valid" symbols, a much richer input language is made available.
3) Whereas handwriting recognition must be customized to a specific language, language-independence is a natural by-product of first-class ink.

With the ink left untranslated, however, new functionality must be developed to replace what is lost by eschewing a standard, character-based representation. For example, the storage requirements of textual ink are much more demanding than ASCII, hence good encoding and compression schemes are needed. And while fast algorithms exist for searching text files (e.g., Unix grep and the "Find" commands in modern word processors), such functionality is not a "given" for ink data.

Adopting this philosophy does not mean that handwriting recognition has no place. It could, in fact, serve as a key underlying technology if the results of recognition are held in the "background" for retrieval purposes and not presented directly to the user. Since there is no opportunity to correct errors (indeed, the whole point is to not bother the user with such concerns), robust search techniques must be employed [20,21]. Figure 2 illustrates this.

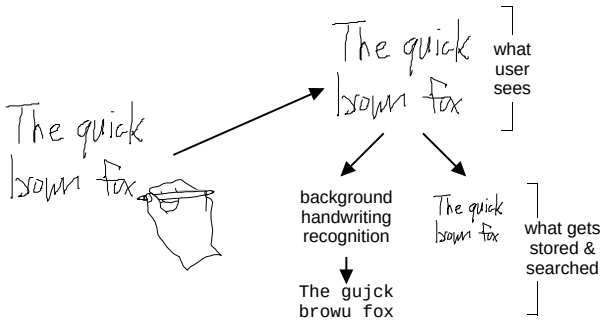

Figure 2: Ink as first-class data.

In this paper, we survey some of the research being done in the area of treating ink as first-class data. Section 2 discusses work on representing, pre-processing, and compressing ink. We review algorithms for retrieving handwritten text and hand-drawn sketches in Sections 3 and 4, respectively. In Section 5, we consider a variety of other related issues, including the use of sketching as an image database query mechanism and systems that associate electronic ink, in the form of annotations, with other kinds of multimedia data. Finally, we offer our conclusions and discuss some open questions in Section 6.

2. REPRESENTING INK DATA

Raw ink data is assumed to be a sequence of points sampled over time, (x,y,t). These record the trajectory of the pen tip over the surface of the digitizer. In addition, "pen-down" and "pen-up" events are noted. Often, the sequence of points between successive pen-down/pen-up's is referred to as a *stroke*. Due to the on-line nature of the problem, the data has both spatial and temporal aspects which can be employed as appropriate.

While several formats for storing ink data have been proposed, including JOT [14] and UNIPEN [12], there is currently no universal standard. Still, the representations used by those working in the field are for the most part equivalent, and converting between them is relatively straightforward.

The raw ink is commonly subjected to a series of pre-processing steps to correct for problems in data acquisition, to compensate for some of the natural variation that arises between writers, and to segment the handwriting into more basic units for later recognition. Depending on the classification stages that follow, the output from pre-processing can be features extracted either from word units or from more primitive character or sub-character units (in the case of textual ink), or from elemental graphics components (in the case of sketches).

The basic stages of pre-processing include segmentation, noise reduction, and normalization. The order in which these are performed can vary from system to system, and specific steps may be repeated and/or skipped altogether. Comprehensive overviews of pre-processing can be found in Tappert et al. [31] and Guerfali and Plamondon [11].

Using textual ink to illustrate, segmentation may take place at three conceptual levels:

1) text line segmentation – divides the input ink data into individual lines,
2) word segmentation – breaks each line into separate word units,
3) character segmentation – if the classifier is character-based, further segments the words into character candidates.

Noise reduction involves the following steps:

1) smoothing – eliminates noise introduced by the tablet or shaky writing,
2) filtering – reduces the number of data points and eliminates "wild" points,
3) de-hooking – removes artifacts ("hooks") that arise at the beginnings and ends of strokes,
4) break correction – eliminates unintended (short) breaks between strokes.

Common normalization procedures include:

1) baseline drift – corrects for the tendency of the text baseline to rise or fall as writing progresses from left to right,
2) writing slant – compensates for the natural slant that can vary widely from writer to writer,
3) size normalization – adjusts the symbols to be a standard size.

Lastly, the pre-processor must generate a representation appropriate for input to the classifier in question. The range of possible representations is enormous and highly dependent on the classifier. A common operation at this point, however, is to segment the input further into sub-character units (a step known as "over-segmentation").

Depending on the rate the pen tip is sampled, and the nature of the writing, the ink data may contain a significant amount of redundancy. In machines with severe memory limitations, such as PDA's, this can lead to inefficient use of the available storage. Several researchers have attempted to address this issue. Chen, et al. describe a new piecewise linear modulation model (including compression results) for handwriting [3]. Wilfong presents on-line algorithms for determining an appropriate subsequence of the sampled points to approximate the curves in question [37].

3. SEARCHING HANDWRITTEN TEXT

Once a body of handwriting has been collected and saved, it is only natural for a user to want to be able to search it. One obvious approach to this problem is to use handwriting recognition to translate the database text and the query into ASCII representations, after which traditional search techniques can be applied. If neither of these steps requires user intervention (i.e., the recognition is performed in the background and the user is not bothered with having to correct errors), the philosophy of treating the ink as first-class data is maintained. Note, though, that this necessitates the retrieval algorithms be tolerant of the kinds of errors that might arise [20,21]. To our knowledge no one has attempted an evaluation of this approach in the domain of handwriting.

In point of fact, ink that is primarily textual can be represented at various levels of abstraction. As depicted in Figure 3, an ink search algorithm could perform approximate matching at any of these levels. A number of researchers have started exploring this idea of comparing the ink

directly [13,16,17,18,28]. While these approaches avoid explicit handwriting recognition, they typically employ a similar notion of elastic matching [30].

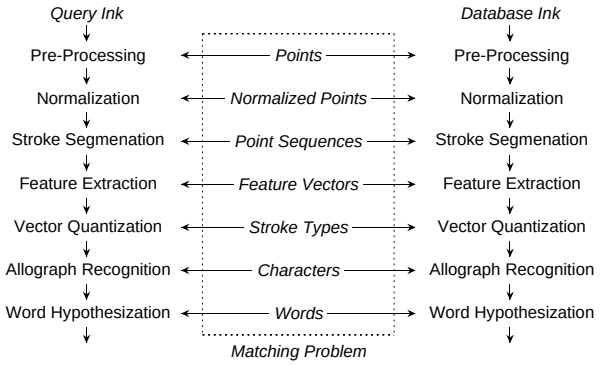

Figure 3: Possible ink matching levels.

We now review a specific algorithm proposed by Lopresti and Tomkins in [18]. Figure 4 shows an overview of ScriptSearch, which consists of four phases. First, the incoming points are grouped into strokes by segmenting the data at local *y*-minima. Next, the strokes are converted into vectors of descriptive features based on those developed by Rubine in the context of gesture recognition [29]. Third, the feature vectors are classified according to writer-specific information using vector quantization, yielding 64 clusters. Finally, the resulting query sequence is matched against each database sequence using approximate string matching over an alphabet of "stroke types."

For this last phase, a well known dynamic programming algorithm is used to determine the "edit distance" between the two sequences [33]. The cost of a deletion or insertion is a function of the amount of ink involved (i.e., the length of the stroke type representing the ink). The cost of a substitution is the distance between the two stroke types in question.

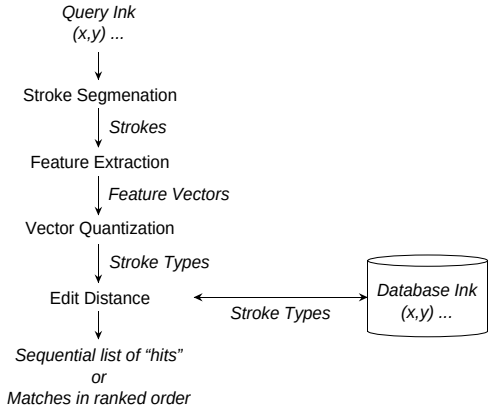

Figure 4: Overview of the ScriptSearch algorithm.

Two new editing operations are also added to the three standard ones: splitting a single stroke into two, and merging two strokes into one. These account for imperfections in the stroke segmentation step. A split/merge table is built that contains information of the form "an average stroke of type α merged with an average stroke of type β results in a stroke of type γ." The cost of splitting a stroke δ into a pair of strokes $\alpha\beta$ is a function of the distance between δ and merge(α,β) = γ. The edit distance is computed using these costs and operations to find matches for the query in the database ink.

Let $dist_{i,j}$ represent the cost of the best match between the first i strokes of the query Q and a substring of the database item D ending at stroke t_j. The recurrence is then:

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

for $1 \le i \le m$, $1 \le j \le n$.

If the decision made at each step in the optimization process is saved, it is possible to backtrack once the computation has completed to recover the precise alignment (or *trace*) between the two sequences. One such example is shown in Figure 5.

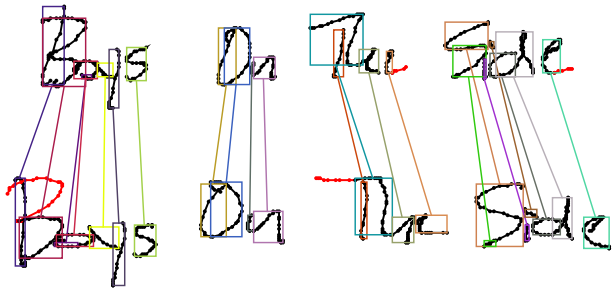

Figure 5: Example of an ink matching trace.

For handwritten text (English and Japanese, cursive and printed), empirical studies using this approach, as well as other, similar ones, have demonstrated good performance. In one experiment reported in [18], two subjects each wrote a reasonably large amount of English text drawn from Herman Melville's famous novel, *Moby Dick*. They then wrote 30 short words and 30 longer phrases (2-3 words), taken from the same passages, which served as search strings.

Two standard criteria from information retrieval were used in assessing the performance of ScriptSearch: *recall*, the percentage of true matches reported, and *precision*, the percentage of reported matches that are true. It is desirable to have both of these measures as close to 1 as possible. There is, however, a fundamental trade-off. By insisting on an exact match, the precision can be made 1, but recall will suffer. On the other hand, if arbitrary edits are allowed between the query and the matched portion of the database, the recall will approach 1, but the precision will fall to 0. For ink to be searchable in this way, there must be a point on the trade-off curve where both recall and precision are sufficiently high.

The chart in Figure 6 presents the results of this experiment, showing precision as a function of recall. The upper set of curves correspond to the longer queries, the lower set to the shorter ones. The bars represent the combined queries for each writer. Note that ScriptSearch returns mostly the desired hits, with relatively little superfluous "noise."

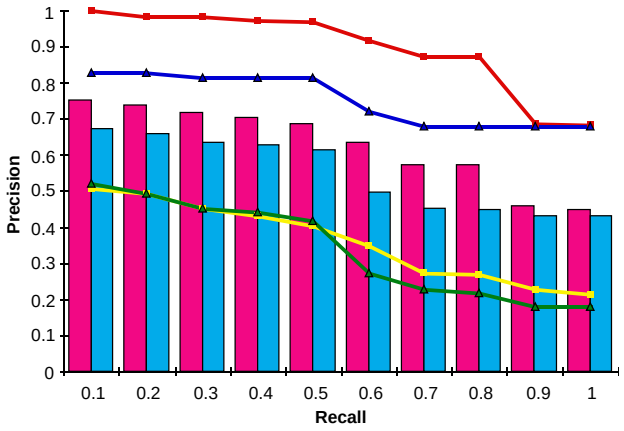

Figure 6: Results of a ScriptSearch experiment.

4. MATCHING HAND-DRAWN SKETCHES

For ink to be truly first-class, a pen-based system must allow for more than just textual input. It should be prepared to handle anything a user would write/draw on a traditional pad of paper.

While the approach described in the previous section works well for writer-dependent matching of textual ink, sketches present more of a challenge as a user might more naturally vary the order in which constituent objects are drawn. Figure 7 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. Algorithms that have been developed for matching textual ink are not flexible enough to capture these kinds of *block motion* in the temporal (stroke-sequence) domain.

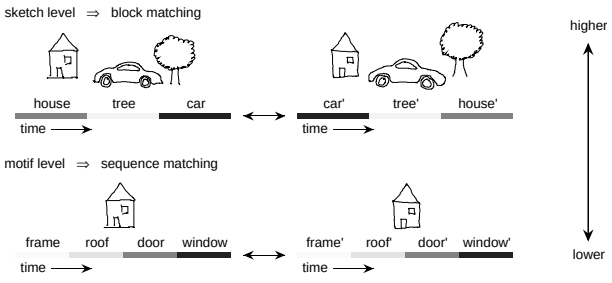

Figure 7: The sketch matching problem.

In a recent paper, Lopresti and Tomkins introduce a family of algorithms for block string matching that address this issue [23]. Here sketch matching is treated at two levels of abstraction. At the highest level, long subsequences of the ink string ("blocks" or "motifs") may be interchanged and repeated to allow for variations in drawing order (e.g., house-tree-car vs. car-tree-house, as demonstrated in the figure). At the lowest level, a single motif is treated as a sequence of strokes that, for a given user, are always drawn in a consistent order, just as in textual ink matching (e.g., frame-roof-door-window). The algorithms for solving this problem, also based on dynamic programming, exploit this structure by partitioning the query ink into blocks adaptively and matching each to a block in the database ink such that the sum of the distances between corresponding blocks is minimized.

Let $Q = q_1q_2...q_m$ and $D = d_1d_2...d_n$ be the query and database ink strings, respectively. A *t-block substring family* of Q, $Q|_t$, is a multiset containing t substrings of Q. The *block edit distance* between two strings 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. The optimization problem is then:

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

While the most general form is NP-complete [23], 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, the following polynomial time algorithm may be employed.

Let $W(i,j)$ be the value of the best possible match between the substring $q_i...q_j$ and any substring of D, plus the per-block cost c_{block}:

$$W(i,j) \equiv c_{block} + \min_{k \le l} \left\{ dist(q_i \mathrm{K}\ q_j, d_k \mathrm{K}\ d_l) \right\}$$

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

$$M(i) = \min_{j<i} \left\{ M(j) + W(j+1, i) \right\}$$

As with textual ink matching, it is possible to reconstruct the best-case alignment between the two ink sequences by saving the optimal decisions at each step and backtracking. An example of such a trace is shown in Figure 8. Here the query sketch depicted on the left side of the figure, a computer and a chair, is matched to a drawing of a chair, a desk, a computer, and a lamp on the right.

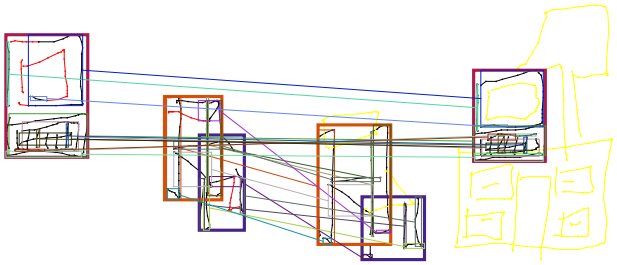

Figure 8: Example of a sketch matching trace.

This approach to sketch matching was tested in an experiment designed so that purely local similarity measures would not be effective [22]. Each of five subjects was asked to create an ink database consisting of 25 sketches, and a set of 25 ink queries to the database. The instruction for creating a database page might be, for example: "Draw a chair, a desk, a computer, and a lamp."

Instructions for creating queries always consisted of 1 or 2 motifs, so a query meant to match the database entry just given might be: "Draw a lamp and a chair." Note that the same query might match other database sketches as well. Of the 25 queries, three matched three entries of the database, nine matched two entries, and 13 matched one entry.

Figure 9 shows the effectiveness of the sketch matching algorithm, measured in terms of the number of false hits that were ranked higher than intended ones. More than 75% of the time, the first match returned was a true hit. Only in a couple cases did the algorithm perform badly.

5. RELATED ISSUES AND APPLICATIONS

As we noted previously, handwriting recognition could serve as a core technology to support the retrieval of first-class ink data. Hence, any improvements that come about as a result of research successes in the field can be used to good advantage. Tappert, at al. present an earlier survey on the state of the art [31]. A more up-to-date paper is one by Plamondon, et al. [27].

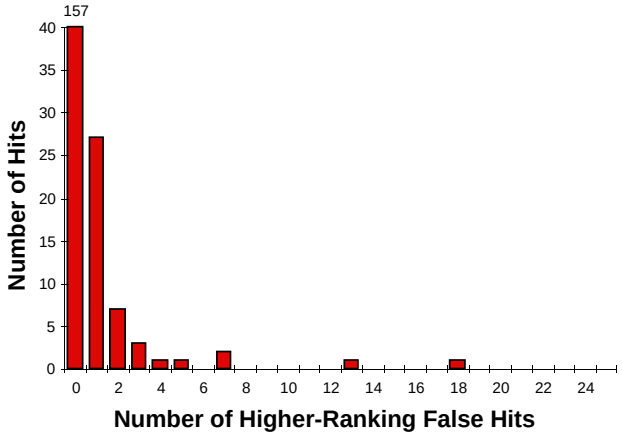

Figure 9: Results of a sketch matching experiment.

Math differs from text in that its logical structure has a second dimension (in addition to the obvious vocabulary differences). A system capable of simultaneously handling text, mathematical expressions, and editing gestures is discussed in Dimitriadis and Coronado [6]. Other systems for handling math, in terms of over-segmenting the input and presenting it to an HMM classifier, are described in Winkler and Lang [38]. A system for segmenting handwritten mathematical expressions based on attributed relational tree representations is presented in Faure and Wang [7].

In terms of graphical input, Welbourn and Whitrow [34] discuss a gesture-based text and diagram editor. The two types of data are identified, with the text presented to one classifier for recognition while the diagrams are beautified by another. Julia and Faure [15] is a paper describing two editors for beautifying drawings that include tables, geometric figures, etc. A pen-based editor for beautifying network diagrams is presented in Galindo and Faure [8]. A strategy for separating annotations from figures in engineering drawings is given in Utton et al. [32]. Tivoli is a pen-based whiteboard application that recognizes and supports various kinds of structures common to meetings, including lists, outlines, and node-link diagrams [25].

Recently, content-based retrieval of image data has become a popular topic for research. Interestingly, even when a keyboard and mouse are available, the quickest, most natural way for people to specify simple shapes is to sketch them. Hence, pen-based input may serve as a front-end tool for image database management. Del Bimbo and Pala describe a technique for this problem based on elastic matching that is scale invariant and takes into account the spatial relationships between objects [5]. Chan, et al. present an approach that extracts prominent geometric features (line segments, circular arcs, and curvelets in general) and uses them to compute a similarity value between the sketch and an image [2].

Another area in which ink shows potential is in combination with other continuous media types. Whittaker, et al. present an application that integrates handwritten notes and recorded audio, allowing users to access particular points in a recording by linking the ink and the audio in the temporal domain [35]. Here, easy-to-browse handwriting provides an index for difficult-to-browse speech. Dynomite is another, similar, such system [36]. Combining writing and speech in asynchronous communication (i.e., messaging) is the subject of a paper by Daly-Jones, et al. [4]. Their studies found that users show a significant preference for pen-and-voice messages over "unimedia" communication.

Beyond the previously cited research efforts, there is growing commercial interest in areas relating to first-class ink data. Microsoft's aha! Inkwriter permits editing of uninterpreted handwritten documents. A. T. Cross's Crosspad product allows users to write on standard paper and then transfer the ink to a PC.

6. CONCLUSIONS AND DISCUSSION

The notion of treating ink as first-class multimedia data offers intriguing possibilities. As we have attempted to show in this short, admittedly incomplete survey, numerous interesting research issues arise when one adopts this view.

Among the important questions that remain unanswered is developing an ability to deal with truly free-form pen input. This would probably involve pre-classifying arbitrary writing/drawing as text, sketches, equations, etc., so that the proper recognition technology can be invoked. As always, it would be best if this could be accomplished without burdening the user in any way.

Given fuzzy retrieval methods such as those described in [20,21], it is not necessary for recognition to be 100% accurate to apply it in the background. Far more important is that it be consistent and robust. In other words, 90% accuracy on all inputs is preferable to 100% accuracy on some inputs and 10% on others. Can we build classifiers with this property? How accurate and robust need they be for acceptable retrieval performance?

On a related note, is it possible to recognize when a given classifier is doing a bad job so that a different technique can be applied or, in the worst case, the user alerted?

Recalling Figure 2, both ink matching and background handwriting recognition have nice properties. The former may be more accurate for a given user, while the latter is writer-independent. Can these two paradigms be combined to yield a more effective retrieval engine?

7. ACKNOWLEDGMENTS

The trademarks mentioned in this paper are the properties of their respective owners.

8. REFERENCES

[1] W. Aref, D. Barbará, D. Lopresti, and A. Tomkins, "Ink as a First-Class Datatype in Multimedia Databases," *Multimedia Database Systems: Issues and Research Directions*, S. Jajodia and V. S. Subrahmanian, editors, Berlin: Springer-Verlag, 1996, pp. 113-163.

[2] Y. Chan, Z. Lei, D. Lopresti, and S. Y. Kung, " A Feature-Based Approach to Retrieval by Sketch," *Proceedings of the SPIE International Symposium on Voice, Video, and Data Communications*, November 1997, pp. 220-231.

[3] H. Chen, O. E. Agazzi, and C. Y. Suen, "Piecewise Linear Modulation Model of Handwriting," *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, August 1997, pp. 363-367.

[4] O. Daly-Jones, A. Monk, D. Frohlich, E. Geelhoed, and S. Loughran, "Multimodal Messages: the Pen and Voice Opportunity," *Interacting with Computers*, vol. 9, no. 1, pp. 1-25.

[5] A. Del Bimbo and P. Pala, "Visual Image Retrieval by Elastic Matching of User Sketches," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, February 1997, pp. 121-132.

[6] Y. A. Dimitriadis and J. L. Coronado, "Towards an Art Based Mathematical Editor that Uses On-Line Handwritten Symbol Recognition," *Pattern Recognition*, vol. 28, no. 6, 1995, pp. 807-822.

[7] C. Faure and Z. X. Wang, "Automatic Perception of the Structure of Handwritten Mathematical Expressions," *Computer Processing of Handwriting*, Singapore: World Scientific, 1990, pp. 337-361.

[8] D. Galindo and C. Faure, "Perceptually-Based Representation of Network Diagrams," *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, August 1997, pp. 352-356.

[9] D. Goldberg and C. Richardson, "Touch-Typing With a Stylus," *Proceedings of the Conference on Human Factors in Computing Systems*, April 1993, pp. 80-87.

[10] M. D. Gross and E. Y.-L. Do, "Ambiguous Intentions: a Paper-like Interface for Creative Design," *Proceedings of the ACM Symposium on User Interface Software and Technology*, November 1996, pp. 183-192.

[11] W. Guerfali and R. Plamondon, "Normalizing and Restoring On-Line Handwriting," *Pattern Recognition*, vol. 26, no. 3, 1993, pp. 419-431.

[12] I. Guyon and L. R. B. Schomaker, "UNIPEN Project of Data Exchange and Recognizer Benchmarks," unpublished standards document, 1993. http://hwr.nici.kun.nl/unipen/.

[13] R. Hull, D. Reynolds, and D. Gupta, "Scribble Matching," *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition*, December 1994, pp. 285-294.

[14] "JOT: A Specification for an Ink Storage and Interchange Format," unpublished standards document, 1993. http://hwr.nick.kun.nl/unipen/jot.html.

[15] L. Julia and C. Faure, "Pattern Recognition and Beautification for a Pen-Based Interface," *Proceedings of the Third International Conference on Document Analysis and Recognition*, August 1995, pp. 58-63.

[16] D. Lopresti and A. Tomkins, "Pictographic Naming," *Adjunct Proceedings of the Conference on Human Factors in Computing Systems*, April 1993, pp. 77-78.

[17] D. Lopresti and A. Tomkins, "Approximate Matching of Hand-Drawn Pictograms," *Proceedings of the Third International Workshop on Frontiers in Handwriting Recognition*, May 1993, pp. 102-111.

[18] D. Lopresti and A. Tomkins, "On the Searchability of Electronic Ink," *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition*, December 1994, pp. 156-164.

[19] D. Lopresti and A. Tomkins, "Temporal-Domain Matching of Hand-Drawn Pictorial Queries," *Handwriting and Drawing Research: Basic and Applied Issues*, M. L. Simner, G. Leedham, and A. J. W. M. Thomassen, editors, Amsterdam: IOS Press, 1996, pp. 387-401.

[20] D. Lopresti and J. Zhou, "Retrieval Strategies for Noisy Text," *Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval*, April 1996, pp. 255-269.

[21] D. Lopresti, "Robust Retrieval of Noisy Text," *Proceedings of the Third Forum on Research and Technology Advances in Digital Libraries*, May 1996, pp. 76-85.

[22] D. Lopresti, A. Tomkins, and J. Zhou, "Algorithms for Matching Hand-Drawn Sketches," *Proceedings of the Fifth International Workshop on Frontiers in Handwriting Recognition*, September 1996, pp. 233-238.

[23] D. Lopresti and A. Tomkins, "Block Edit Models for Approximate String Matching," *Theoretical Computer Science*, vol. 181, 1997, pp. 159-179.

[24] D. Lopresti, M. Y. Ma, P. S. P. Wang, and J. D. Crisman, "Ink Matching of Cursive Chinese Handwritten Annotations," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 12, no. 1, 1998, pp. 119-141.

[25] T. P. Moran, P. Chiu, W. van Melle, and G. Kurtenbach, "Implicit Structures for Pen-Based Systems within a Freeform Interaction Paradigm," *Proceedings of the Conference on Human Factors in Computing Systems*, 1995, pp. 487-494.

[26] M. Nakagawa, K. Machii, N. Kato, and T. Souya, "Lazy Recognition as a Principle of Pen Interfaces," *Adjunct Proceedings of the Conference on Human Factors in Computing Systems*, April 1993, pp. 89-90.

[27] R. Plamondon, D. P. Lopresti, L. R. B. Shomaker, and R. Srihari, "On-Line Handwriting Recognition," *Wiley Encyclopedia of Electrical and Electronic Engineering*, to appear.

[28] A. Poon, K Weber, and T. Cass, "Scribbler: A Tool for Searching Digital Ink," *Human Factors in Computing Systems Conference Companion*, May 1995, pp. 252-253.

[29] D. Rubine, *The Automatic Recognition of Gestures*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, 1991.

[30] C. C. Tappert, "Cursive Script Recognition by Elastic Matching," *IBM Journal of Research and*

Development, vol. 26, no. 2, November 1982, pp. 765-771.

[31] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The State of the Art in On-Line Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 8, pp. 179-190.

[32] G. Utton, M. Cripps, D. G. Elliman and C. A. Higgins, "A Strategy for On-Line Interpretation of Sketched Engineering Drawings," *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, August 1997, pp. 771-775.

[33] R. A. Wagner and M. J. Fischer, "The String-to-String Correction Problem," *Journal of the Association for Computing Machinery*, vol. 21, no. 1, 1974, pp. 168-173.

[34] L. K. Welbourn and R. J. Whitrow, "A Gesture Based Text and Diagram Editor," *Computer Processing of Handwriting*, Singapore: World Scientific, 1990, pp. 221-234.

[35] S. Whittaker, P. Hyland, and M. Wiley, "Filochart: Handwritten Notes Provide Access to Recorded Conversations," *Proceedings of the Conference on Human Factors in Computing Systems*, April 1994, pp. 271-277.

[36] L. D. Wilcox, B. N. Schilit, and N. Sawhney, "Dynomite: A Dynamically Organized Ink and Audio Notebook," *Proceedings of the Conference on Human Factors in Computing Systems*, March 1997, pp. 186-193.

[37] G. Wilfong, "On-Line Algorithms for Compressing Planar Curves," *Proceedings of the Symposium on Discrete Algorithms*, 1997, pp. 158-165.

[38] H. J. Winkler and M. Lang, "Symbol Segmentation and Recognition for Understanding Handwritten Mathematical Expressions," *Proceedings of the Fifth International Workshop on Frontiers in Handwriting Recognition*, September 1996, pp. 465-469.