EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2006)
Thomas Stahovich and Mario Costa Sousa (Editors)

Automatic interpretation of proofreading sketches

J.A. Rodríguez, G. Sánchez and J. Lladós

Computer Vision Center (Computer Science Department, Universitat Autonoma de Barcelona, Spain)

Abstract

We present a sketch-based system for proofreading documents. The gestures and words drawn by the proofreader on a document view are translated into high-level actions that represent editions such as replace, insert, delete and others. Our particular system is not restricted to a predefined alphabet of gestures. Instead, any symbol can be employed for striking out words or drawing inserts. This provides more flexibility and adaptability to the user. In contrast to other similar works, our interface integrates interpretation and recognition of handwritten words. The described system has been implemented for proofreading digital documents on screen but also for paper documents printed on Anoto paper and annotated using a digital pen.

Categories and subject descriptors: I.7.5 [Document capture]: Graphics recognition and interpretation, Document analysis.

1. Introduction

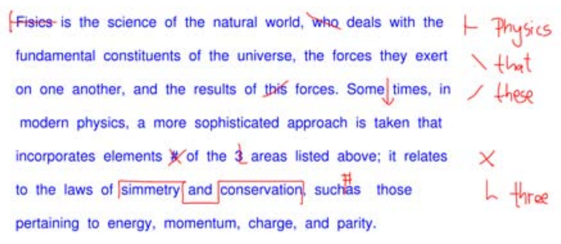
Despite the exponential increase of use of computers in many aspects of the daily life, there are still many usual situations in which the computer appears as a non-ergonomic and technically complex device. We still roll back to pen and paper in situations like fast sketching of an idea, taking notes or annotating documents.

Some digital devices exist that provide a natural pen interface not achievable by keyboard and other usual input peripherals. These devices such as palmtop and tablet computers, pen mice or digital pen and paper, allow systems accepting gestures [Rub91] and handwriting [PS00] as input. These devices open new possibilities such as document edition or proofreading based on sketches.

There are previous works that address this problem using various approaches. The automatic correction of printed documents was already addressed in [MB97] but from the image-based point of view. The advantage of this system is that the correction can be done with pen on paper, the most natural interface, but lacks of the on-line counterpart. The same authors have proposed an on-line version [BGM97] but it is based on menus and not on sketches. In [HKB93] a sketch-based system for on-line text editing is presented but it only supports delete, insert and move operations. Each of the actions is denoted by a specific gesture. This system only allows basic edition with restricted gestures. Another sketch-based proofreading system with a set of 11 gestures is reported in [AR99]. Even if the set of gestures is wider, there are a few items that break the naturality provided by hand movement: words must be inserted by typing and the user must click after finishing each gesture. Finally, a number of other pen-based system exist for annotating documents without or with little recognition capability, like [BM03, SW04], and even systems [Gui03] using emerging devices such as Anoto [Ano03] paper and digital pen .

In this paper we present a sketch-based system for proofreading documents. Our system interprets the sketched marks and words and translates them into high-level edition actions, such as replace, insert, delete, swap words, etc. It is possible, at the end of the process, to generate a new version of the document that is updated with the editions of the proofreader. Our system presents some advantages with respect to the discussed previous works. First, it is able to recognize handwritten input, determining the word label and associating it to the correct annotation. Second, our system has been implemented for tablet computers but also for proofreading documents printed on Anoto paper. The couple of Anoto paper and digital pen results in an emerging device that closes the loop between paper and digital documents. Finally, we let the user the possibility of striking out a word with any symbol rather than with a specific one. This makes the problem more complex but provides an open, usable and intuitive interface.

The rest of the paper is structured as follows. In Section 2 the sketch alphabet of our system is presented, which allows to formulate the problem formally in Section 3. Then, in Section 4 the designed system is described in detail. In Section 5 we discuss the results of some tests of performance. Finally, in Section 6 the conclusions are commented.

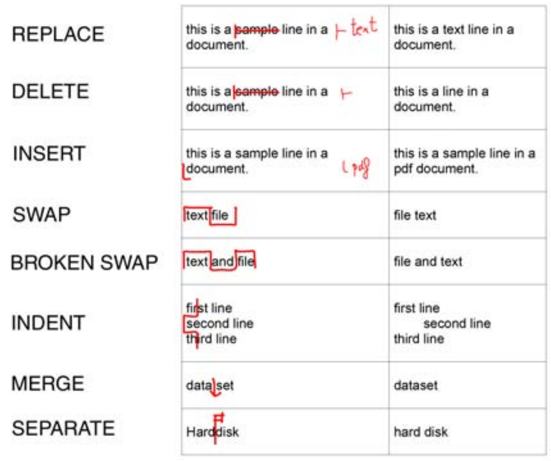

~~Fisics~~ is the science of the natural world, ~~who~~ deals with the fundamental constituents of the universe, the forces they exert on one another, and the results of ~~this~~ forces. Some times, in modern physics, a more sophisticated approach is taken that incorporates elements ~~of~~ of the ~~3~~ areas listed above; it relates to the laws of simmetry and conservation, suchas those pertaining to energy, momentum, charge, and parity.

Physics
that
these
three

Figure 1: *Document excerpt annotated by a proofreader using a sketching interface*

2. Sketch alphabet

Our automatic proofreading interpreter accepts sketches drawn using the typographic conventions used by proofreading professionals. In Figure 1 a sample text fragment is shown. The proofreading sketches on this text follow the rules summarized in Figure 2. These rules provide gestures for the following actions: replace a text by another text, delete text, insert text, swap words, swap non-contiguous words, indent a line, merge two words and split a word in two.



REPLACE	this is a ~~sample~~ line in a document. text	this is a text line in a document.
DELETE	this is a ~~sample~~ line in a document.	this is a line in a document.
INSERT	this is a sample line in a document. pdf	this is a sample line in a pdf document.
SWAP	text file	file text
BROKEN SWAP	text and file	file and text
INDENT	first line second line third line	first line second line third line
MERGE	data set	dataset
SEPARATE	Harddisk	hard disk

Figure 2: *Proofreading notation used by the presented system.*

On the one hand, there is a set of actions (SWAP, BROKEN SWAP, INDENT, MERGE and SEPARATE) that we will call *direct* actions since they are directly encoded by a specific gesture. On the other hand, the actions REPLACE, DELETE and INSERT need annotations at the margin. For instance, to replace some word by another, the proofreader must strike out the word with a symbol. Then this symbol is repeated at the page margin (normally in reduced size) and finally the correct word is handwritten next to the repeated symbol. For deleting, the process is the same except that no text is written at the margin. Inserting works analogue to replace, with the obvious difference that instead of striking out a word, a symbol is placed at the position where the new text should be appended.

One novelty with respect to existing systems is the possibility of specifying these strike outs or insert symbols for REPLACE, DELETE and INSERT using any symbol. We take advantage from the fact that they appear in pairs to allow a more open and intuitive interface. The advantages of using an unrestricted alphabet of gestures for these cases are:

- The necessary previous knowledge of the particular system is reduced.
- The proofreader is more flexible to use preferred or improvised symbols.
- Different symbols can be used if the same edition action has to be repeated in a different part of the text, so that there is no possibility for confusion (in fact, this is a "best practice" in some proofreading conventions).

3. Formulation of the problem

From a formal viewpoint, proofreading consists of encoding a set of actions that affect the document contents into a set of sketches that can be either gestures or text. Let us denote the set of actions

$$A = \{A_k\} \tag{1}$$

with $k = 1 \dots N_A$ where N_A is the number of actions. Each action A_k is an action from the following alphabet

$$A_k \in \{\text{REPLACE, DELETE, INSERT, SWAP BROKEN SWAP, INDENT, MERGE, SEPARATE}\} \tag{2}$$

The sketches on the document are represented by a set of strokes

$$S = \{S_i\}, i = 1 \dots N_S \tag{3}$$

with N_S the number of strokes in S. Each stroke can be described by the set of points sampled by the sketching device between each pen down and pen up events:

$$S_i = \{\vec{p}_j\}, \vec{p}_j = (x_j, y_j), j = 1 \dots N_{S_i} \tag{4}$$

or, alternatively, by the segments that link these points:

$$S_i = \{s_j\}, s_j = \text{seg}(\vec{p}_{j-1}, \vec{p}_j), j = 2, \dots N_{S_i} \tag{5}$$

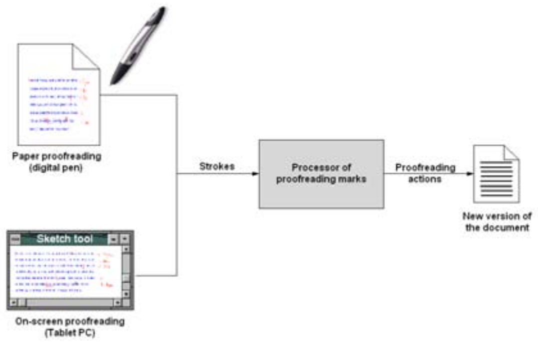

Paper proofreading (digital pen)
Sketch tool
On-screen proofreading (Tablet PC)
Strokes
Processor of proofreading marks
Proofreading actions
New version of the document

Figure 3: *System overview*

where N_{S_i} is the number of points sampled in stroke S_i and seg$(\vec{p},\vec{q})$ designs the segment that starts at point $\vec{p}$ and ends at point $\vec{q}$. The automatic proofreading interpretation process can be formalized as finding the decoding f that converts S to A

$$A = f(S) \tag{6}$$

As the document sketches are processed as a whole and not one by one, this decoding implies a coupling of the following problems:

- Symbol recognition
- Handwriting recognition
- Layout analysis

Of special interest is the problem of symbol recognition. As discussed in section 2, one can find two categories of symbols. On the one hand, there are the symbols that appear in pairs and that belong to an unrestricted alphabet that will be called W_u. On the other hand, there will be the symbols whose shape provides a direct meaning. These symbols are from a restricted symbol alphabet denoted W_r from now on.

Additionally, in the set of direct symbols there are symbols that have a fixed shape (SEPARATE, MERGE) and some others whose shape will vary depending on the words on which they act.

The specific details about the whole automatic proofreading system can be read in the following sections.

4. System architecture

Figure 3 shows an overview of the system. A user annotates either a printed document with digital pen or a PDF document on-screen, obeying the syntax explained in Section 2. A proofreading engine interprets the annotations and extracts all the contents affected by the corrections and the actions to apply to these contents. From this output it is possible to refactor the annotated document, obtaining an updated version where the output edition actions have been applied.

The system can be logically divided into input, processing and output blocks. The processing block can be further subdivided into a layout segmentation module, a symbol recognition module and a handwriting recognition module.

The system must be robust to multiple annotations in the same paragraph end even in the same line, and to multiple line text annotations; and must support the different symbols from W_u that the proofreader might write but at the same time it must permit sketching the actions with the same symbol class if they are clearly separated.

The implementation of the system has been developed with the Java programming language, supported by a Java platform for handling digital ink that has been programmed by the authors.

4.1. Input

The input of the system is the set of all strokes drawn by a corrector with a sketching device. As stated previously, we implemented the possibility of annotating using a digital pen on PDF documents printed on paper enabling Anoto functionality. And we also designed an on-screen editor which accepts mouse strokes. This possibility becomes useful for tablet computers or when using a pen mouse. In any case, the stroke format is unified and the coordinates refer to the document sheet. This is a device independent approach that allows future extension of the input possibilities.

4.2. Layout segmentation

The layout segmentation module is devoted to the analysis of the spatial information. Instead of trying to directly recognize symbols or groups of symbols and assigning them to one of the categories in Figure 2, one must take into account that the particular problem we are trying to solve is extremely context-dependent. So it is more interesting to first try to extract the highest possible amount of context information. Then, the recognition step will be conducted more effectively according to this context information (this can help e.g. in selecting the appropriate recognition method, discarding some classes, etc.)

As the context is encoded in the spatial information this step can be called layout segmentation. In this step, the strokes are grouped into symbols, they are then categorized into text and margin symbols and finally blocks of symbols that will be processed as a whole in subsequent modules are found.

The first step is to group the strokes into symbols using a connected component labelling. Two strokes S and S' are considered to be connected if

$$\exists i,j \text{ such that } \sqrt{(x_i - x'_j)^2 + (y_i - y'_j)^2} < d_{TC} \tag{7}$$

where d_{TC} is a threshold distance that is empirically determined. For a more realistic connectivity determination, additional points were linearly interpolated for each stroke when evaluating the expression in Equation 7.

Once a symbol $\mathcal{S}$ has been obtained from a set of connected strokes, it is categorized into the group of symbols on the text and the group of symbols on the margin (shortly, text symbols and margin symbols) with the following criterion:

$$\mathcal{S} \in \begin{cases} \text{TEXT} & x_{min} < X_{MARGIN} \\ \text{MARGIN} & \text{otherwise} \end{cases}, x_{min} = \min_{(x_i,y_i)\in\mathcal{S}} x_i \tag{8}$$

where X_{MARGIN} is the x-coordinate on the document where the margin starts at.

The final step in the layout segmentation is to divide the paper vertically into blocks that contain annotations that are close to each other. We experienced that a high frequency of annotations in the same paper zone tends to produce blocks of margin symbols (shortly, margin blocks). If $(x_{BB}, y_{BB}), (x'_{BB}, y'_{BB})$ are the points defining the bounding box of a block, then the text zone with a vertical coordinate y such that $y_{BB} < y < y'_{BB}$ is called the influence zone of this block. In Figure 4 we present an example of a situation with two annotation blocks and their influence zones.

This figure illustrates how finding margin blocks and their influence zones simplifies the processing. On the one hand, if a text symbol $\mathcal{S}$ is not on any influence zone, we will consider that $\mathcal{S} \in W_p$. On the other hand, the the symbols $\mathcal{S}_a$ in a block such that $\mathcal{S}_a \in W_u$ must have a matching text symbol, which is expected to be in the influence zone.

The procedure to build blocks is also a connectivity labelling. Two symbols are considered to be connected if the vertical distance between their bounding box center is below a threshold d_{TB}. The influence zone of each symbol is determined as already explained.

Processing the annotation blocks separately rather than the whole document is advantageous since it reduces the number of matches to evaluate in further steps and thus the probability of confusion.

4.3. Symbol recognition

A number of elaborate techniques exist for recognizing symbols using different strategies [LVSM02]. However, with the layout already segmented, the sketches on the document can be decoded into actions taking much advantage of the available context, relying less on the symbol recognition itself. The strategy to follow is: first, from each influence zone the text symbols that match with the margin symbols are found. Then, the remaining symbols and the symbols that are not inside any influence zone are recognized using a particular symbol classifier.

Search for matching symbols

First, we subdivide each of the margin blocks into lines, using again a labelling of connected components. For this purpose we consider that two symbols belong to the same line if their bounding box center is separated by a distance shorter than a threshold d_{TL}, where, obviously $d_{TS} < d_{TL} < d_{TB}$. This labelling results in a set of lines $L = L_1 \dots L_{N_L}$ where each line L_i contains a set of symbols $L_i = \{\mathcal{S}_j\}$. From each line L_i, the symbol with smallest horizontal projection is extracted:

$$\mathcal{S}_i^{left} = \arg\min_{\forall j} x_{BB}(\mathcal{S}_j) \tag{9}$$

where $x_{BB}(\mathcal{S})$ stands for the x-coordinate of the $\mathcal{S}$-bounding box. Notice from Figure X. that each $\mathcal{S}_i^{left}$ can be either a symbol $\mathcal{S}_i^{left} \in W_u$ or $\mathcal{S}_i^{left} \in H$

Denote $\mathcal{S}_k$ the set of symbols on the influence zone of a block and $d(\mathcal{S}, \mathcal{S}')$ some distance measure between symbols $\mathcal{S}$ and $\mathcal{S}'$. Then a matrix D is built where each element is computed as

$$D_{ki} = d(\mathcal{S}_k, \mathcal{S}_i^{left}) \tag{10}$$

We search for the lowest distance value $D_{k'i'}$ in matrix D. The indices (k', i') are stored in a list and columns i and j are removed from matrix D. The process of finding the lowest distance is repeated until all remaining distance values in the matrix are below a threshold d_M, or until there are no more columns or rows in the matrix. At this moment we have a list of index pairs (k', i') that indicate a REPLACE, DELETE or INSERT action coded by $\mathcal{S}_k$ on the text and its pair $\mathcal{S}_i^{left}$ at the page margin.

The rest of the symbols in line L_i are considered to be words (or pieces of words). If in the consecutive following line L_{i+1} it happens that $\mathcal{S}_{i+1}^{left}$ had no match with any $\mathcal{S}_k$ then it is considered to be the continuation of text from the previous line, and so with every consecutive line. If the number of word symbols is greater than 0, we have decoded a REPLACE or an INSERT action. Both actions are distinguished from whether the involved symbol S_k intersects some printed words or not. In case that the number of words at the margin is 0, then we encounter a DELETE action. For each action, the complete set of words will be send to the handwriting recognition module preserving their spatial relations.

If a block only contains one line and its influence zone only contains one text symbol (lowest annotation in Figure 4), the correspondence is directly done and the matching process with matrix D is omitted.

Matching with Hausdorff distance

The symbol matching is performed using the line segment Hausdorff Distance [GL02]. This is a generalization of the Hausdorff distance between two point sets for computing distances between segment sets.

Let us define a distance between two segments s and s', which takes into account the angle between the segments,

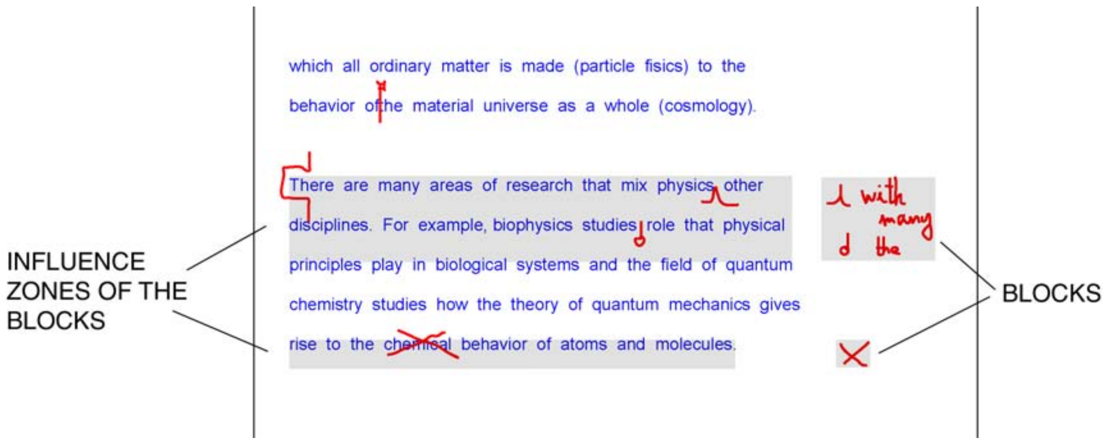


Figure 4: *Concept of annotation blocks. The symbols in the margin are clustered into blocks of similar position. The zone in the text overlapping each block represents the influence zone. The influence zone defines the region where the potential matches of the annotation symbols are.*

and the distance in horizontal and vertical projections:

$$d(s,s') = \sqrt{W^2 d_\theta(s,s')^2 + d_{\parallel}^2(s,s') + d_{\perp}^2(s,s')} \qquad (11)$$

where W is a parameter that is empirically determined, $d_\theta(s,s')$ is the tangent of the rotation angle for aligning the shortest segment with the longest one, and $d_{\parallel}$ and $d_{\perp}$ are the respective distances in parallel and normal direction with the shortest segment already aligned.

With this definition, the line segment Hausdorff distance between two sets of segments S and T is defined as

$$H(S,T) = max(h(S,T), h(T,S)) \qquad (12)$$

where

$$h(S,T) = \frac{1}{\sum_{s_i \in S} l_{s_i}} \sum_{s_i \in S} l_{s_i} \min_{t_j \in T} d(s_i, t_j) \qquad (13)$$

with l_{s_i} the length of segment s_i. Equation 13 is called directed distance and it is an improvement of the expression used in the classical Hausdorff distance.

A symbol is a set of strokes that can be represented as segments (Equation 5). If two symbols $\mathcal{S}$ and $\mathcal{T}$ are aligned, then $H(\mathcal{S},\mathcal{T})$ can be interpreted as a dissimilarity measure. For computing the dissimilarity or, simply, distance between two symbols, first their coordinates will be scaled and shifted to a predefined bounding box and then equation 12 will be applied.

Predefined symbol recognition

Text symbols without matching margin annotations and text symbols outside influence zones of blocks are classified into the set W_p of predefined symbols using a two-stage symbol classifier.

First, note that the some segments of the SWAP and BROKEN SWAP symbols have different lengths as a function of the words they involve. Moreover, it is allowed that both symbols appear reflected. The selected strategy is first to try to classify a symbol using the directional information of the sketches to match predefined templates; and for imperfect or unrecognized symbols apply a secondary classifier based on the symbol shape.

In the first stage the strokes of the symbol to recognize are approximated to straight segments using the method described in [HN04]. The angles of the resulting segments are quantized in one of eight possible directions (N, NE, E, SE, S, SW, W, NW) and a chain is build from these direction codes. The direction codes are matched against predefined templates with a certain tolerance.

For the imperfect strokes that do not match the predefined templates, a Zernike moment descriptor classification using support vector machines is performed using library hhreco [HN04]. A previous training process is necessary for this classifier. One of the advantages of Zernike moments is that they are invariant to rotations and it is possible to detect the rotated versions of the SWAP symbols and even rotations of other symbols due to imperfect sketching.

The final Zernike moment classifier is also able to reject doubtful samples so that no proofreading decision is taken at all for very suspect symbols.

4.4. Handwritten word recognition

A handwriting recognition module is necessary to translate the words written by the proofreader at the margin. Several methods [PS00] as well as commercial engines are available. For this particular application a commercial module that supports on-line cursive handwriting has been used.

4.5. System output

A list of proofreading actions is generated as a result of the interpretation method. The actions include on which text to apply and, when necessary, which new text is attached. From this action list we can generate a new version of the document which includes all modifications that come from the proofreading interpretation. In the next section, some examples will be shown.

5. Experiments

The experiment sections shows both evaluation experiments used to design the system and evaluation of the system performance.

5.1. Evaluation of matching measures

In 4.3 the line Hausdorff distance was introduced. Our system uses this dissimilarity measure for the matching mechanism. The decision for the Hausdorff distance is taken after the study that we present now.

On a dataset of 700 samples from a single writer we have computed a modified Dunn's index [BP98] for different distances. The modified Dunn index gives an idea of the separability of classes given a distance measure between elements:

$$\nu = \min_{1 \le s \le c} \left\{ \min_{1 \le t \le < c} \left\{ \frac{\delta(X_s, X_t)}{\max_{1 \le k \le c} \Delta(X_k)} \right\} \right\} \quad (14)$$

where $\delta(X_s, X_t)$ represents some distance between classes X_s and X_t and $\Delta(X_k)$ is a definition of the diameter of class X_k. We have taken

$$\delta(S,T) = \frac{1}{N_S N_T} \sum_{x \in S, y \in T} d(x,y) \quad (15)$$

and

$$\Delta(S) = \frac{1}{N_S(N_S - 1)} \sum_{x,y \in S, x \neq y} d(x,y) \quad (16)$$

where N_S is the number of elements in class S and d is the evaluated distance measure. Higher values of the modified Dunn index indicate more class separability for distance d.

In table 5.1 we present the Dunn index computed for different distance measures. As you can appreciate there, the measure with highest performance among the tested ones is the line segment Hausdorff distance.

The Line Segment Hausdorff distance appears as a very useful distance method between symbols, especially suited for on-line recognition as the extraction of the segments from the strokes is natural.

5.2. Performance of the automatic system

This section describes the experiments carried out to measure the performance of the automatic proofreading interpreter. In these experiments, users were given a text document with some errors and were told to proofread it obeying the explained notation. The sketches of the users were done with digital pen. For this purpose, the PDF document to annotate was printed on digital paper enabling Anoto functionality. One of the samples is presented in figure 5. This benchmark resulted in a set of 33 documents filled out by 9 different proofreaders.

The most important point in our system is the association of annotations in the margin with their corresponding text symbol. It is clear that it is not the main target of this work to develop a symbol or a text recognizer. Therefore, we concentrate on measuring the degree of correctness with which the similar symbol pairs are matched and the annotation text correctly associated, and also the cases where this is not accomplished and for which reasons. Therefore, in each of the 33 outputs, we examine all annotations that have margin contents (REPLACE, DELETE and INSERT) and classify them in one of the following cases:

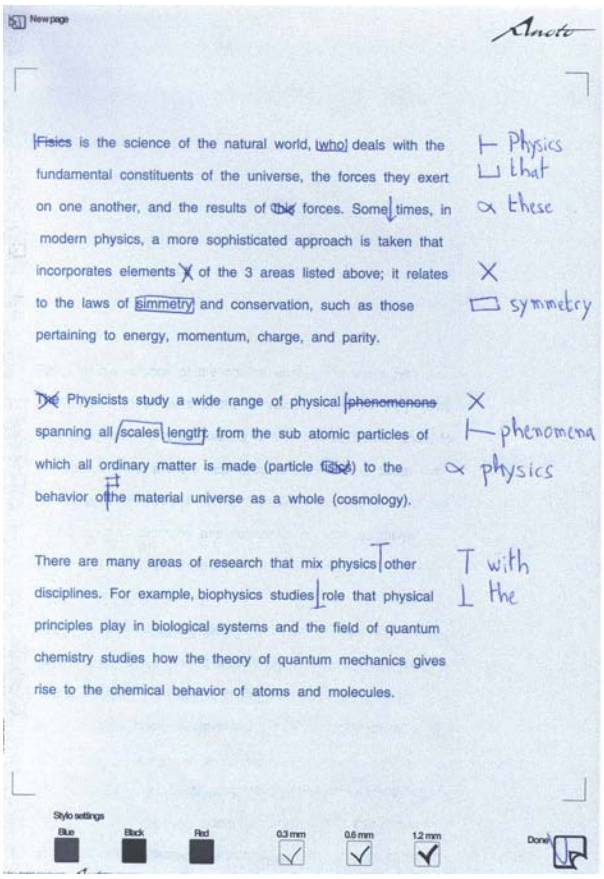
- Correct association: The text symbol has been associated

Distance	ν
Line segment Hausdorff distance (W=0.25)	1.73
ED between Geometric moments (L=1)	1.10
ED between Line moments [LN96] (L=1)	1.01
ED between Zernike moments (L=13)	0.87
ED between Line Legendre moments [LN96] (L=1)	0.74
ED between Legendre moments (L=1)	0.66

Table 1: *Modified Dunn indices computed for different distance measurements. ED stands for Euclidean distance. W is the parameter of the Hausdorff distance and L is the moment order. Only the best result for each distance among all the parameter variations is shown.*

Position	Action	Replaced text	New Text
0	REPLACE	Fisics	Physics
50	REPLACE	who	that
188	REPLACE	this (188)	these
203	MERGE		
316	DELETE	#	
386	REPLACE	simmetry	symmetry
492	DELETE	The	
546	REPLACE	phenomenons	phenomena
581	SWAP		
679	REPLACE	fisics	physics
711	SEPARATE		
721	INSERT		with
877	INSERT		the

Table 2: *List of actions obtained from the automatic proofreading interpreter for the previous sample. Position is an index for locating the word characters inside the document contents.*



New page
Anoto

~~Fisics~~ is the science of the natural world, who deals with the fundamental constituents of the universe, the forces they exert on one another, and the results of this forces. Some times, in modern physics, a more sophisticated approach is taken that incorporates elements of the 3 areas listed above; it relates to the laws of simmetry and conservation, such as those pertaining to energy, momentum, charge, and parity.

The Physicists study a wide range of physical ~~phenomenons~~ spanning all scales length from the sub atomic particles of which all ordinary matter is made (particle fisics) to the behavior ofthe material universe as a whole (cosmology).

There are many areas of research that mix physics other disciplines. For example, biophysics studies role that physical principles play in biological systems and the field of quantum chemistry studies how the theory of quantum mechanics gives rise to the chemical behavior of atoms and molecules.

Physics
that
these
symmetry
phenomena
physics
with
the

Figure 5: *Document annotated by a proofreader. The PDF document was printed on digital paper and the proofreading marks were sketched with digital pen.*

Correct associations	90.1 %
Incorrect associations	4.5 %
Missing associations	5.4 %
Number of documents with 100% correct actions	22

Table 3: *Results of the evaluation of correct association of annotations with text at the margin.*

with the corresponding margin symbol and the annotation text has been associated correctly.

- Incorrect association: The text symbol has been associated with a non-corresponding margin symbol, or the text has not been correctly association.
- Missing action: The text symbol is not associated with any symbol and has been processed as a symbol from W_p.

Results are summarized in table 3.

It can be observed that 90.1% of these annotations are correctly associated. Recall that for correctly associating annotations first a correct layout segmentation is necessary and then symbol pairs extraction from the set of all-with-all possible pairs, where additionally the symbol form is unrestricted, so that 90.1% is a good result for this first prototype. It should be noted that this 90.1% correctness is not a regular observation in most documents. On the contrary, it is due to few samples with unacceptable result. Observe in Table 3 that 22 out of the 33 documents had 100% correct associations (these are document containing an average of 6.5 annotations with margin, the smallest with 4 and the highest with 11 annotations with margin).

Regarding the reos of actions (direct actions), 83% of the symbols were correctly recognized, with 8.4% errors and 4.6% rejections. This not so high recognition rate may be due to the fact that the SVM classifier was trained with samples from a single writer. A recent experiment with the symbols extracted from the 33 documents, using the half for training and the half for test, increased the recognition rate to 95.0% with 5% error rate and 0% rejection rate.

From the analysis of the incorrect samples we observe that in most cases it happened that imperfect sketching increased the Hausdorff distance between symbols and prevented them from being matched correctly. At this point we believe that it would be of great help to have a method for discrimination between symbols and words. The additional information provided by this system for each symbol could help in the match search process and would increase the 90.1% accuracy.

6. Conclusions

We have presented a system that automatically interprets digital ink annotations that proofreaders sketch on documents. The system successfully recognizes the set of proposed gestures and recognizes the associated handwritten notes, something that is not usual in similar works. Moreover, a part of the actions can be specified using gestures from an unrestricted alphabet. With this, we have generalized the problem and have performed a step towards the ideal system that interprets any document annotation.

The Line Segment Hausdorff distance appears as a very useful dissimilarity measure between aligned sketched symbols. This distance behaves robustly to imperfect writing of the symbols.

For improving the performance of the automatic interpreter a text/symbol discriminator could be implemented for being applied to each of the symbols at the margin. This would provide supporting information just in the layer between layout segmentation and symbol recognition.

Under the performance conditions of such a system, a validation screen would be helpful for the proofreader to accept the interpreted actions. If the proofreader is also enabled to perform changes in the validation screen, this information could serve as new ground truth for retraining the processes involved in the refactoring.

Acknowledgements

This work has been partially supported by the Spanish project CICYT TIC2003-09291.

References

[Ano03] ANOTO: Development guide for services enabled by Anoto functionality, 2003.

[AR99] ANDRÉ J., RICHY H.: Paper-less editing and proofreading of electronic documents. In *EuroTeX '99 Proceedings* (1999).

[BGM97] BUNKE H., GONIN R., MOERI D.: A tool for versatile and user-friendly document correction. In *Proceedings of the 4th International Conference on Document Analysis and Recognition* (1997), IEEE Computer Society, pp. 433–438.

[BM03] BARGERON D., MOSCOVICH T.: Reflowing digital ink annotations. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (2003), ACM Press, pp. 385–393.

[BP98] BEZDEK J. C., PAL N. R.: Some new indexes of cluster validity. *IEEE Transactions on Systems, Man and Cybernetics 28*, 3 (June 1998), 301–315.

[GL02] GAO Y., LEUNG M. K. H.: Line segment Hausdorff distance on face matching. *Pattern Recognition 35*, 2 (2002), 361–371.

[Gui03] GUIMBRETIÈRE F.: Paper augmented digital documents. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology* (2003), ACM Press, pp. 51–60.

[HKB93] HARDOCK G., KURTENBACH G., BUXTON W.: A marking based interface for collaborative writing. In *UIST '93: Proceedings of the 6th annual ACM symposium on User interface software and technology* (1993), ACM Press, pp. 259–266.

[HN04] HSE H., NEWTON A. R.: Sketched symbol recognition using Zernike moments. *Proceedings of the 17th international conference on pattern recognition (ICPR'04) 01* (2004), 367–370.

[LN96] LAMBERT G., NOLL J.: Discrimination properties of invariants using the line moments of vectorized contours. In *Proceedings of ICPR'96* (1996), pp. 735–739.

[LVSM02] LLADÓS J., VALVENY E., SÁNCHEZ G., MARTÍ E.: Symbol recognition: Current advances and perspectives. In *GREC '01: Selected Papers from the Fourth International Workshop on Graphics Recognition Algorithms and Applications* (2002), Springer-Verlag, pp. 104–127.

[MB97] MÖRI D., BUNKE H.: *Automatic interpretation and execution of manual corrections on text documents*. Handbook of Character Recognition and Document Image Analysis. World Scientific, 1997, pp. 679–702.

[PS00] PLAMONDON R., SRIHARI S. N.: On-line and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22* (2000), 63–82.

[Rub91] RUBINE D.: Specifying gestures by example. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (1991), ACM Press, pp. 329–337.

[SW04] SHILMAN M., WEI Z.: Recognizing freeform digital ink annotations. In *Document Analysis Systems* (2004), pp. 322–331.