A SIGN MATCHING TECHNIQUE TO SUPPORT SEARCHES IN SIGN LANGUAGE TEXTS

Antônio Carlos da Rocha Costa, Graçaliz Pereira Dimuro, Juliano Baldez de Freitas

ESIN/UCPel – Escola de Informática
Universidade Católica de Pelotas
{rocha,liz,jubafreitas}@atlas.ucpel.tche.br

Abstract

This paper presents a technique for matching two signs written in the `SignWriting` system. We have defined such technique to support procedures for searching in sign language texts that were written in that writing system. Given the graphical nature of `SignWriting`, a graphical pattern matching method is needed, which can deal in controlled ways with the small graphical variations writers can introduce in the graphical forms of the signs, when they write them. The technique we present builds on a so-called degree of graphical similarity between signs, allowing for a sort of "fuzzy" graphical pattern matching procedure for written signs.

1. Introduction

For the most part, software for processing sign language texts and databases have started to be developed only recently, simultaneously with the spreading of interest in `SWML` (Costa, 2003) among software developers concerned with the `SignWriting` (Sutton, a; Sutton, c) system. Obviously, an important and critical operation needed for such sign language processors is that of searching signs in sign language texts.

This paper presents a technique for matching two signs written in the `SignWriting` system. We have defined such technique to support procedures for searching sign language texts that were written in that writing system. Given the graphical nature of `SignWriting`, a graphical pattern matching method is needed, which can deal in controlled ways with the small graphical variations writers can introduce in the graphical forms of signs when they write them. The technique we present builds on a so-called degree of graphical similarity between signs, allowing for a sort of "fuzzy" graphical pattern matching procedure for written signs.

The paper is organized as follows. In section 2., we review aspects of sign languages related to the problem of having them written in some notation, and summarize the main features of the `SignWriting` system. Section 3. summarizes the work done on `SWML` and its importance for the development of software for processing `SignWriting` texts and databases. Section 4. presents the main contribution of the paper, namely, the sign matching technique designed to support procedures for searching in sign language texts. Section 5. brings the Conclusion. The sample signs presented in the paper are from the Brazilian sign language LIBRAS (Linguagem Brasileira de Sinais).

2. Sign languages and the `SignWriting` system

Along history, no writing system has been widely established for sign languages, so that such languages have always been used only for face-to-face communication.

Since Stokoe, in the 1960's, first recognized that sign languages are full natural languages, in the same sense that oral languages are, some notation systems for sign languages have been proposed. Stokoe himself introduced one such notation system (W. C. Stokoe and Croneberg, 1976). HamNosys (Hanke,) was another proposal. Both were conceived as technical tools for registering linguistic features of sign languages (handshapes, movements, articulation points, etc.).

`SignWriting` is also a proposed system for writing sign languages (Sutton, a). Contrary to the other systems, however, which were proposed mainly as tools for technical linguistic work, `SignWriting` was proposed as tool for daily use, by common (Deaf) people (Sutton, b).

3. `SignWriting` and `SWML`

Both the Stokoe system and HamNoSys are based on a linear representation of signs, using special characters for such purpose. `SignWriting` is based on graphical, bi-dimensional representations, using graphical symbols.

This way, the former systems can easily be encoded in computers in a linear way, by simply assigning numeric codes to each special character, and the technique for searching signs in texts written with such systems should be straight forward to develop.

`SignWriting`, on the other hand, requires that, besides the numeric encoding of each symbol, the computer representation of a sign keeps the information concerning the relative position of each symbol in the bi-dimensional area occupied by the representation of the sign (this complicates the searching procedure, as is shown below).

The `SignWriter` program (Sutton et al., 1995), the first computer editor for sign languages, defined such an encoding for `SignWriting`. That encoding was a binary encoding, created specifically for the needs of that program.

`SWML` (Costa, 2003) is a proposal for a general encoding format for `SignWriting` documents, using `XML` (**?**). It builds on the encoding used by the `SignWriter` program, presenting it in a fashion the makes such encoding available for use in all kinds of computer applications of `SignWriting` (document storage and retrieval, on-line dictionaries, computer interpretation and generation of sign languages, etc.). The `SW-Edit` program (Torchelsen et al., 2002) fully relies on `SWML` to store `SignWriting`-based sign language texts. `SignWriting` and `SWML` were proposed (Costa and Dimuro, 2002; Costa and Dimuro, 2003) as foundations for Sign Language Processing, the transposition of the methods and techniques of Natural Language Processing and Computational Linguistics, that have long been developed for oral language texts, to sign language texts.

The rest of this paper tackles one of the simplest operation one can do on a sign language document, namely, searching for a specific sign.

4. Matching Written Signs

There is a particular problem that has to be solved to allow sound searching procedures for sign languages files written in SignWriting, namely, to define a way of dealing with the small graphical variations that writers can introuce in the forms of the signs, when they write them.

The `SignWriting` system distinguishes explicitly some graphical properties of the symbols of a sign, like rotation and flop, for example, but does not distinguish tiny variations due to vertical and/or horizontal displacements of symbols within the sign, because such values are allowed to vary along the whole range of available positions within a sign box (as opposed to, e.g., rotation, which can only assume a small set of possible discrete values). The consequence of having such a "continuous" set of possible positions of symbols within a sign_box is that one lacks a clear geometric definition for the similarity between two signs, if they differ only with respect to the positions of their corresponding symbols.

The solution we have found to that problem is to allow the user to control the criteria to be used for judging on the degree of similarity of two signs by giving him a means to define a "fuzzy" correspondence between the component symbols of the two signs. The resulting matching procedure guarantees that two corresponding symbols have the same symbol type, rotation and flop, but allows them to have the (user specified) degree of variation on their relative positions within the respective signs instances. This kind of similarity between two signs is formalized in this section as a parameterized, reflexive and symmetric relation, that we call *sign similarity relation.*

4.1. Basic Geometric Features of Symbols and Signs

Initially, we formalize the basic geometric information concerning `SignWriting` symbols and signs.

Definition 1 *A* symbol s *is defined as a tuple* $s = (c, n, f, v)$*, where the values of* c, n, f *and* v *vary according to the symbol set being used, and:*

(i) c *is the category number (not available in symbol sets previous to the SSS-2002 symbol set (Sutton, c); use* $c = 0$ *in such cases),*

(ii) n *is the shape number (within the symbol's category),*

(iii) v *is the symbol variation (a complimentary information distinguishing symbols by features like, e.g., if the index finger is curved or not, in the symbol for the index handshape),*

(iv) f *is the filling information (encoding, e.g., palm orientation, in a symbol for a hand).*

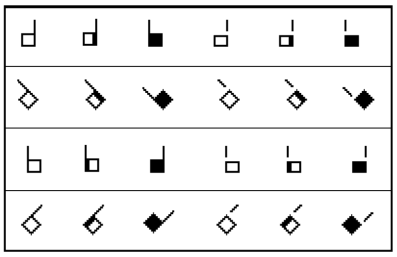

Figure 1: The group $G_{0/0}$ of symbols called *index*, and some of its rotated and flopped elements.

A set of symbols having the same symbol category and shape (c, n) and differing only in their filling or variation information, is called a *symbol group*, denoted by $G_{c/n}$. For each symbol group $G_{c/n}$ there is a so-called *basic symbol*, denoted by $s_{c/n}$, for which $f = 0$ and $v = 0$, so that $s_{c/n} = (c, n, 0, 0)$.

Definition 2 *An* oriented symbol ***S*** *is defined as a tuple* ***S*** $= (s, r, fp)$*, where:*

(i) s *is a symbol of any symbol group* $G_{c/n}$*,*

(ii) r *indicates the (counter clockwise) rotation operation applied to* s*, relative to the basic symbol* $s_{c/n}$ *of the symbol group* $G_{c/n}$ *(the rotation is given in intervals of 45 degrees, for all symbols sets available up to now), and*

(iii) fp*, called flop, is a Boolean value indicating if the symbol* s *is vertically mirrored or not, relative to the basic symbol* $s_{c/n}$ *of the symbol group* $G_{c/n}$*.*

Example 1 *The symbol group called* index*, denoted by* $G_{0/0}$*, whose symbols, with category* $c = 0$ *and shape* $n = 0$*, represent hands with index finger straight up and closed fist, is shown in Figure 1. Each symbol* s *in the group* $G_{0/0}$ *is a tuple* $s = (0, 0, 0, f)$*, with variation* $v = 0$ *and fill* $f = 0, 1, ..., 5$ *(from left to right in the figure). The oriented symbols in the first row have the basic orientation (no rotations, no flop) and are given by tuples of the form* ***S*** $= (s, 0, 0)$*. Each different fill information is represented by a different color fill in the symbol, indicating a different palm orientation, starting with the palm oriented towards the signer's face. In the second row, a rotation of 45 degrees was applied to each symbol, and the oriented symbols in that line are thus given by* ***S*** $= (s, 1, 0)$*. In the third and fourth rows, representing the left hand, there are flopped symbols, given by* ***S*** $= (s, 0, 1)$ *(with no rotations) and* ***S*** $= (s, 7, 1)$ *(with rotations of 315 degrees).*

Definition 3 (i) *A* symbol box *is the least box that contains a symbol, defined as the 4-uple* $sb =$

(x, y, w_{sb}, h_{sb}), where x and y are, respectively, the horizontal and vertical coordinates of the upper left corner of the symbol box (relative to the upper left corner of the sign box containing the symbol box — see item (iv)*), w_{sb} is its width and h_{sb} is its height;*

(ii) *A* symbol instance*, that is, an occurrence of an oriented symbol within a sign, is defined as a pair **Si** $=$ (**S**; sb), where **S** $= (s, r, fl)$ is an oriented symbol and sb is its symbol box;*

(iii) *A* sign*, denoted by **Sg**, is a finite set of symbol instances;*

(iv) *A* sign box *is a box that contains a sign, defined as a pair $Sgb = (w_{Sgb}, h_{Sgb})$, where w_{Sgb} is the box width and h_{Sgb} is the box height;*

(v) *A* sign instance *is defined as a tuple **Sgi** $=$ (**Sg**; Sgb; p), representing a sign **Sg** together with a sign box Sgb that contains it, and an index p indicating the position of the sign instance within the sign sequence (sign phrase) to which it belongs .*

All the definitions presented above are reflected in the SWML format. Note, in particular, that as defined above, sign boxes (and consequently, sign instances) have no coordinate information. This is so because sign language texts should be conceived essentially as strings of signs, with no particular formatting information included in them.

SWML, however, defines the notions of *document*, *page*, *line* and *cell*, so that sign instances can be put into cells, sequences of cells organized into lines, sequences of lines into pages, and sequences of pages into documents, in order to support document rendering procedures (e.g., horizontal or vertical renderings). Note also that symbols don't have predefined sizes (width and height). Sizes are defined only for symbol instances, through their symbol boxes. This allows for scalable symbol sets (e.g., in the SVG format (**?**)).

Example 2 *The* SWML *representation of the LIBRAS sign for* IDEA *(written as in Figure 2) is:*

```
<signbox>
 <symb x="46" y="37" x-flop="0" y-flop="0"
    color="0,0,0">
   <category>04</category>
   <group>02</group>
   <symbnum>001</symbnum>
   <variation>01</variation>
   <fill>01</fill>
   <rotation>04</rotation>
 </symb>
 <symb x="81" y="48" x-flop="0" y-flop="0"
    color="0,0,0">
   <category>01</category>
   <group>01</group>
   <symbnum>001</symbnum>
   <variation>01</variation>
   <fill>02</fill>
   <rotation>02</rotation>
 </symb>
 <symb x="62" y="18" x-flop="0" y-flop="0"
    color="0,0,0">
   <category>02</category>
   <group>01</group>
   <symbnum>001</symbnum>
   <variation>01</variation>
   <fill>01</fill>
   <rotation>01</rotation>
 </symb>
 <symb x="99" y="31" x-flop="0" y-flop="1"
    color="0,0,0">
   <category>02</category>
   <group>05</group>
   <symbnum>001</symbnum>
   <variation>01</variation>
   <fill>01</fill>
   <rotation>02</rotation>
 </symb>
</signbox>
```

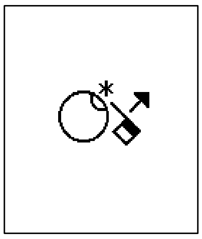
Figure 2: A way to write the LIBRAS sign for IDEA.

4.2. The Sign Similarity Relation

The *sign similarity relation* is a parameterized, reflexive, symmetric and non transitive relation, introduced here to formalize the approximate similarity between two sign instances, and to provide for the construction of matching procedures for signs and sign language expressions.

The sign similarity relation has to embody an admissible difference in the positions of corresponding symbol instances within the two sign instances that it relates, taking into account a measure of significance for this difference, as determined by the user. The admissible differences in the positions of corresponding symbol instances are expressed in terms of percentages of some reference sizes, by a so-called *minimum degree of correspondence*, denoted by ε.

The reference sizes may be given either explicitly (e.g., 10 pixels) or implicitly (e.g., as the height and width of some symbol instance, chosen for that purpose among the symbols of the symbol set).

More over, the admissible difference in the corresponding positions of the corresponding symbols may be calculated in two ways:

- with respect to their *absolute* positions within the sign boxes to which they belong
- with respect to their positions *relative* to some reference symbol, known to be instantiated in each of the signs being compared

The *absolute* way of calculating the admissible differences is simpler, but the *relative* way allows the establishment of the similarity between a sign and another deriving

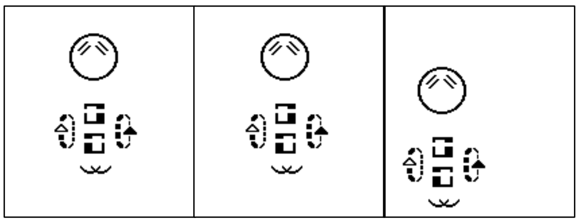

Figure 3: Similarity based on absolute and relative positions of the symbols (LIBRAS sign for YEAR).

from it just by a joint displacement of the symbols within the sign_box: e.g., in figure 3, the first sign instance would usually be judged similar only to the second instance, according to an absolute position based similarity relation, while it could also be judged similar to the third instance, according to the relative position based similarity relation.

We now define the sign similarity relation based on the absolute positions of the symbols.

Definition 4 *Let* $\boldsymbol{Si}_1 = (\boldsymbol{S}_1; sb_1)$ *and* $\boldsymbol{Si}_2 = (\boldsymbol{S}_2; sb_2)$ *be two symbol instances belonging to two different signs. Let their symbol boxes be given by* $sb_1 = (x_1, y_1, w_{sb1}, h_{sb1})$ *and* $sb_2 = (x_2, y_2, w_{sb2}, h_{sb2})$*, respectively. Then,* $\boldsymbol{Si}_1$ *and* $\boldsymbol{Si}_2$ *are said to* correspond to each other with at least degree ε*, and reference sizes* h_0 *and* w_0 *(for height and width), denoted by* $\boldsymbol{Si}_1 \approx^{\varepsilon}_{h_0,w_0} \boldsymbol{Si}_2$*, if and only if the following conditions hold:*

(i) Equality between the basic symbols*:*
$\boldsymbol{S}_1 = \boldsymbol{S}_2$ *(which implies* $w_{sb1} = w_{sb2}$ *and* $h_{sb1} = h_{sb2}$*),*

(ii) Admissible horizontal difference*:*
$|\frac{x_1 - x_2}{w_0}| \leq k$

(iii) Admissible vertical difference*:*
$|\frac{y_1 - y_2}{h_0}| \leq k$

where $k = \frac{100-\varepsilon}{100} \geq 0$.

Definition 5 *Let* $\boldsymbol{Sgi}_1 = (\boldsymbol{Sg}_1; Sgb_1; j_1)$ *and* $\boldsymbol{Sgi}_2 = (\boldsymbol{Sg}_2; Sgb_2; j_2)$ *be two sign instances.* $\boldsymbol{Sgi}_1$ *and* $\boldsymbol{Sgi}_2$ *are said to be* similar with at least degree ε*, relative to the absolute positions of their symbols, and reference sizes* h_0 *and* w_0*, if and only each symbol in a sign has one and only one corresponding symbol in the other sign, that is, there exists a bijection* $f : \boldsymbol{Sg}_1 \rightarrow \boldsymbol{Sg}_2$*, such that for each* $\boldsymbol{Si} \in \boldsymbol{Sg}_1$*,* $\boldsymbol{Si} \approx^{\varepsilon}_{h_0,w_0} f(\boldsymbol{Si})$.

Example 3 *Consider the three instances of the LIBRAS sign* IDEA *which are in Figure 4. Observe that each such sign instance contains an instance of the symbol* index *which differs in its coordinates from the corresponding* index *symbol instance of the other sign instances (all other symbol instances match exactly their correspondents). Consider a situation where a user is searching for that sign* IDEA *in a text. Suppose he writes the first sign instance as the sign to be searched and that only the two other instances are present in the text. The later two instances have some degree of similarity with the first sign instance. In spite of this fact, they are graphically different from the first instance, in a strict sense. They may all be considered to represent the same sign, or not, depending on the minimum degree of similarity required by the user for the results of the matching procedure. If the user specifies an intermediate degree of similarity, the second instance would match the first, while the third instance would not (the hand is too low in comparison with its position in the first sign instance). If the user specifies a low degree of similarity, all instances would match. If the user required 100% of similarity, no instance would match. The total degree of similarity (*$\varepsilon = 100\%$*) requires that no difference be admitted between the two sign instances being compared.*

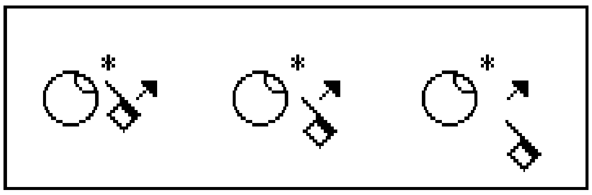

Figure 4: Three (possible) instances of the LIBRAS sign *IDEA*.

The basic similarity relation defined above does not take into account some important (and frequent) exceptions. Such exceptions are mainly related to symbols like the *arrow* symbol (encountered, e.g., in the LIBRAS sign IDEA), whose position within the sign is, in general, not critical (see Figure 5). Such symbols have most of their meaning completely encoded in their shapes and transformations, and the place where they are put in the sign boxes is essentially irrelevant. For instance, the *arrow* symbol in the sign for IDEA means that the right hand moves in the horizontal plane, in the indicated direction, and this information is the same, wherever the *arrow* is placed in the sign box. In such cases, the relative position of the symbol within the sign box is not important. In the examples of the Figure 5, even if a rigorous or a total degree of similarity is required, the matching process should find that those three sign instances are similar. On the other hand, for symbols like the *asterisk*, almost no variation of the its position should be allowed, since it indicates a position where two components of the sign (e.g., head, hands, etc.) touch each other when the sign is performed, and even small degrees of variations may imply linguistically relevant differences between the signs.

Other reasonable definitions for the sign similarity relation could be given such as, for instance, the one already mentioned, of taking the positions of the symbols relatively to a reference symbol, known to occur on both the sign instances that are being compared. Even coarser relations could be defined, and possibly considered useful, e.g., one defining the admissible differences on the basis of the absolute coordinates of the very symbols being compared.

4.3. Search Procedures for Sign Texts

`SWML`, as currently defined, already has all information

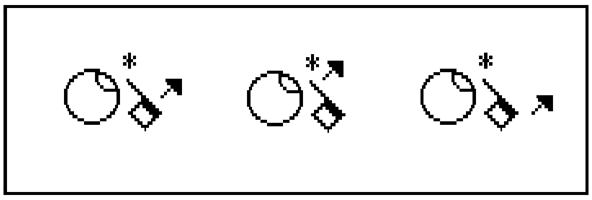

Figure 5: Three (guaranteed) instances of the LIBRAS sign *IDEA*.

needed to allow for a *sign matching procedure* based on the sign similarity relation defined here. The special treatment of symbols whose meanings are not sensitive to the symbols' placements in the signs is to be embedded in the matching process, requiring from SWML only that it identifies symbol instances completely, which it perfectly does. On the basis of such sign matching procedure, a procedure to search for signs in sign language texts can be easily defined, in a straightforward way.

5. Conclusion

In this paper, we have shown that searching for signs in sign language texts written in SignWriting is a straight forward matter. The only slightly tricky part of the searching procedure is in the operation of matching two signs, which should allow for small differences in the positions of their corresponding symbol instances. Ideally, the size of the differences that are to be admitted in such correspondence tests should be specifiable by the user when he calls the search procedure, so that he can have full control over the desired degree of similarity of the signs being compared.

Acknowledgements

This work was partially supported by CNPq and FAPERGS.

6. References

A. C. R. Costa and G. P. Dimuro. 2002. SignWriting-based sign language processing. In I. Wachsmuth and T. Sowa, editors, *Gesture and Sign Language in Human-Computer Interaction*, pages 202–05, Berlin. Springer-Verlag.

A. C. R. Costa and G. P. Dimuro. 2003. SignWriting and SWML: Paving the way to sign language processing. In O. Streiter, editor, *Traitement Automatique des Langues de Signes, Workshop on Minority Languages, Batz-sur-Mer, June 11-14, 2003*.

A. C. R. Costa. 2003. The SWML site. Located at:. `http://swml.ucpel.tche.br`.

T. Hanke. Hamnosys - the hamburg notation system. Located at:.

V. Sutton. Lessons in signwriting – textbook and workbook. Available on-line at:. (Sutton, c).

V. Sutton. The signwriting history. Available on-line at:. (Sutton, c).

V. Sutton. The SignWriting site. Located at:. `http://www.signwriting.org`.

V. Sutton, , and R. Gleaves. 1995. *SignWriter – The world's first sign language processor*. Center for Sutton Movement Writing, La Jolla.

R. P. Torchelsen, A. C. R. Costa, and G. P. Dimuro. 2002. Editor para textos escritos em signwriting. In *5th. Simposium on Human Factors in Computer Systems, IHC 2003*, pages 363–66, Fortaleza. SBC.

D. C. Casterline W. C. Stokoe and C. G. Croneberg. 1976. *A Dictionary of American Sign Language on Linguistic Principles*. Linstok Press, Silver Spring.