

Collana Studi e Ricerche 50

SCIENZE E TECNOLOGIE

Informatisation of a graphic form of Sign Languages

Application to SignWriting

Fabrizio Borgia



SAPIENZA
UNIVERSITÀ EDITRICE

2016

Copyright © 2016

Sapienza Università Editrice

Piazzale Aldo Moro 5 – 00185 Roma

www.editricesapienza.it

editrice.sapienza@uniroma1.it

ISBN 978-88-9377-004-0

Iscrizione Registro Operatori Comunicazione n. 11420

La traduzione, l'adattamento totale o parziale, la riproduzione con qualsiasi mezzo (compresi microfilm, film, fotocopie), nonché la memorizzazione elettronica, sono riservati per tutti i Paesi. L'editore è a disposizione degli aventi diritto con i quali non è stato possibile comunicare, per eventuali involontarie omissioni o inesattezze nella citazione delle fonti e/o delle foto.

All Rights Reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording or any other information storage and retrieval system, without prior permission in writing from the publisher. All eligible parties, if not previously approached, can ask directly the publisher in case of unintentional omissions or incorrect quotes of sources and/or photos.

Cover image: Fabrizio Borgia, *SignVectors* (2016)

"And they shall know no fear."
The Emperor of Mankind

Contents

Abstract	xxiii
Introduction	1
1. Deafness and communication	5
1.1. Deafness and language	5
1.1.1. Sign language in history	5
1.1.2. Sign Language and written representation	15
1.1.2.1. Glosses	17
1.1.2.2. The Stokoe notation	18
1.1.2.3. The HamNoSys notation	19
1.1.2.4. The SignWriting notation	20
1.1.2.5. A comparison of notations for Sign Language	21
1.2. Deafness and the digital divide	24
2. SignWriting	29
2.1. The history of SignWriting	30
2.2. Overview of SignWriting system	38
2.3. A different SignWriting coding	47
2.4. Digital SignWriting	48
2.4.1. Digital resources in SignWriting	48
2.4.1.1. Websites	48
2.4.1.2. Blogs	49
2.4.1.3. Wikipedia	49
2.4.1.4. Digital editors	50

2.4.2. Challenges of digital SignWriting	51
3. SWord	53
3.1. The SWord framework	53
3.2. SWift	56
3.2.1. Introduction to SWift	56
3.2.2. Core features	58
3.2.2.1. Interface design	59
3.2.2.2. Glyph search system	61
3.2.2.3. Support for user-defined glyphs	63
3.2.2.4. Toolbox design	65
3.2.2.5. Saving options and formats	65
3.2.2.6. Storyboard	66
3.2.3. Usability test	66
3.2.3.1. Methodology	66
3.2.3.2. Results	74
3.2.3.3. Discussion	77
3.2.3.4. Limitations	79
3.2.4. After the test	79
4. Introduction to SW-OGR	81
4.1. Reasons to work for SW-OGR	81
4.2. Related work on OCR	86
4.2.1. Data acquisition	91
4.2.2. Pre-processing	91
4.2.3. Segmentation	92
4.2.4. Feature extraction	95
4.2.5. Classification	97
4.2.6. Post-processing	100
4.3. SW-OGR issues at a glance	101
5. Coding system of SW-OGR	105
5.1. SignWriting datasets	105
5.2. Core recognition elements	108
5.3. OGR coding for SignWriting	113
5.3.1. Hand Contacts (Co)	118
5.3.2. Hand Configurations (HaC)	124

5.4. Interoperability between coding systems	135
6. Implementation of SW-OGR	139
6.1. Architecture of SW-OGR	139
6.1.1. Concept	139
6.1.2. Software architecture	141
6.2. Design and development of SW-OGR	157
6.2.1. Image pre-processing	158
6.2.2. Image binarization	159
6.2.2.1. Thresholding methods	159
6.2.2.2. Thresholding performance evaluation	160
6.2.3. Shape detection	169
6.2.3.1. Circles	169
6.2.3.2. Rectangles, triangles and other polygons	175
6.2.4. OGR area inference	177
6.2.4.1. Head Circles and Contacts (HeE_HE)	178
6.2.4.2. Hand Configurations (HaC)	182
6.2.5. Glyph recognition	183
6.2.5.1. Head Circles and Contacts (HeE_HE)	183
6.2.5.2. Hand Configurations (HaC)	184
6.3. Development status and limitations	196
7. Testing of SW-OGR	199
7.1. Testing of SW-OGR	199
7.1.1. Head Circles and Contacts (HeE_HE)	200
7.1.2. Facial Expressions: Eyes (HeE_EY)	202
7.1.3. Hand Configurations (HaC)	205
7.2. The SW-OGR Engine in action	207
Conclusions and future research	211
Acknowledgments	235

List of Figures

1.1.	<i>Death of St. Bernardine</i> , painted by Pinturicchio in the church of St. Mary in Aracoeli.	9
1.2.	Excerpt from John Bulwer's <i>Chirologia</i> (Bulwer, 1644), presenting a very early sign dictionary.	11
1.3.	ASL sign for <i>bear</i> , glossed in English. Images extracted from (Sutton, 1996).	18
1.4.	ASL sign for <i>bear</i> , written using the Stokoe notation. Images extracted from (Sutton, 1996).	19
1.5.	ASL sign for <i>bear</i> , written using the HamNoSys notation. Images extracted from (Sutton, 1996).	20
1.6.	ASL sign for <i>bear</i> , written using the SignWriting notation. Images extracted from (Sutton, 1996).	21
1.7.	ASL sign for <i>bear</i> , written using different notations. Images extracted from (Sutton, 1996).	21
1.8.	A comparison between writing systems, conducted taking into account different factors. Image extracted from <i>ASL Font: Ways to write ASL</i> (2013)	22
1.9.	A group of hand configurations in use within ASL, expressed using different writing systems. Image extracted from <i>ASL Font: Ways to write ASL</i> (2013)	24
2.1.	LIS sign for <i>Fun</i> , written in SignWriting. Extracted from (Di Renzo et al., 2012)	30

- 2.2. First steps of the *Sugar Plum Fairy*, from the second act of the *Nutcracker* ballet, recorded with DanceWriting (Sutton, 1983). 31
- 2.3. Early fragment of SignWriting (1976), extracted from (Sutton & von der Lieth, 1976). This fragment represents a Danish SL sentence meaning *It is father*. The significant influence of DanceWriting over the newborn SignWriting can be identified by the glyph shapes and by the presence of a three-lined staff. 32
- 2.4. Detail of the ASL Wikipedia page about Abraham Lincoln, written in SignWriting. 33
- 2.5. Communities which use SignWriting around the world (Sutton, 1996). 36
- 2.6. Difference between expressive and receptive point of view. Extracted and translated from (Di Renzo et al., 2012). 39
- 2.7. LIS sign for *Fun*, written in SignWriting. Different colors highlight glyphs belonging to different ISWA categories. 41
- 2.8. Base symbol of a hand configuration featuring a single index finger extended. Extracted from (Sutton, 1996) 45
- 2.9. Orientation variations available for a glyph belonging to Category 1 - Hands. Extracted from (Sutton, 1996) 45
- 2.10. Plane variations available for a glyph belonging to Category 1 - Hands. Extracted from (Sutton, 1996) 45
- 2.11. Rotation variations available for a glyph belonging to Category 1 - Hands. Original configuration image extracted from (Sutton, 1996) 46
- 2.12. Illustration of the 13-digit Sutton ISWA code used to uniquely identify any SignWriting glyph. 46
- 2.13. An example of the improvements introduced by ISWABIANCHINI. The white boxes show the possible trajectories of hand movements within ISWA2008, arranged by geometric planes; The orange boxes show the missing trajectories which were added to improve the system consistence. 48
- 2.14. Home screen of the *SignWriting Website* (available at <http://www.signwriting.org/>). 49

- | | |
|--|----|
| 2.15. Home screen of the <i>Frost Village</i> blog (available at http://www.frostvillage.com/). | 50 |
| 2.16. Home screen of the <i>ASL Wikipedia Project</i> (available at http://ase.wikipedia.wmflabs.org/wiki/Main_Page). A subset of the available entries are visible. | 50 |
| 2.17. Screenshot of <i>SWift</i> SignWriting digital editor (available at http://visel.di.uniroma1.it/SWift/). | 51 |
| 3.1. Concept illustration of the <i>SWord</i> framework (courtesy of Prof. Maria De Marsico). | 55 |
| 3.2. The user interface of Sutton's <i>SignMaker</i> (Sutton, 1996). | 57 |
| 3.3. <i>SWift</i> 's (v. 1.01) home screen, divided in 4 functional areas. | 60 |
| 3.4. Search interface related to the anatomic area of the hands. | 62 |
| 3.5. Search interface related to the anatomic area of the hands: one criterion (glyphs involving only three extended fingers) is selected in one of the choose boxes. | 63 |
| 3.6. Support for handwritten glyphs provided by <i>SWift</i> : the glyph is first handwritten (left) and then imported on the sign display (right). | 64 |
| 3.7. Animation frames of two buttons within the interface of <i>SWift</i> : <i>Anti-clockwise Rotation</i> and <i>Delete</i> . | 65 |
| 3.8. Storyboard: the story composition screen. The management buttons (open, save, delete) are visible in the title bar (top) and the signs already added are visible in the center of the screen, along with the buttons for their management (create, edit, delete). | 67 |
| 3.9. Spatial setting for the Think by Signs Test. | 69 |
| 3.10. VL representation of task number 6 of the Think by Signs Test. | 71 |
| 3.11. Web page showing a question of the QUIZ employed for the test. | 73 |
| 3.12. Difference between the puppet of V.1.00 (before the usability test) on the right and the puppet of V.1.01 (after the test) on the left; the circle with the hand icon highlights the mouse pointer. | 79 |
| 4.1. Component diagram for a new generation of SignWriting editors featuring a <i>SW-OGR</i> Engine. | 82 |

4.2.	Component diagram for a new generation of mass Sign-Writing corpora digitalizer featuring a SW-OGR Engine.	85
4.3.	Arab letters consist of strokes and dots. If they are present, dots can occur from 1 to 3 times (top). Moreover, dots can be above, in the middle or below the letter (bottom).	87
4.4.	Different baseline alignments in different notations. From left to right: the alphabetic baseline (Arabic, Latin, etc.), hanging baseline (Indian scripts, etc.), ideographic baseline (East Asian ideograms, etc.).	89
5.1.	Sample of the DS-PEAR dataset.	106
5.2.	Sample of the DS-POITIERS dataset.	107
5.3.	Sample of the DS-DEV dataset.	108
5.4.	An example of text belonging to the DS-POITIERS dataset.	109
5.5.	An example of slice (courtesy of C.S. Bianchini). This is the first slice extracted from the text in Fig. 5.4.	110
5.6.	An example of frag (courtesy of C.S. Bianchini). This is the first frag extracted from the slice in Fig. 5.5.	111
5.7.	The complex relationship between frag and glyphs: a frag may represent a set of glyphs (left), a glyph (middle) or just a part of a glyph (right - the glyph is composed by three different frags).	111
5.8.	Set of frags detected within the text in Fig. 5.4. Each frag has been highlighted with a different random color.	112
5.9.	A few examples of the features encoded by an OGR identifier: from left to right: presence of geometric shapes, presence of convexity defects within the convex hull, presence of foreground clusters.	114
5.10.	Glyph representing the shape of an eye (ISWA-2010: 04-02-006-01-01-01). The original digital version is visible on the left, along with some examples of drawing inaccuracies, found within our datasets.	115
5.11.	The whole set of glyphs within the Hand Contacts area (85 symbols).	119
5.12.	Glyphs within the Hand Contacts area, without rotations (25 symbols).	120
5.13.	Organization of a OGR tree.	122

5.14. Hand Contacts area after the <i>base shape</i> check.	122
5.15. Hand Contacts area after the <i>repetition</i> check. A number of internal nodes and leaf nodes are hidden for the sake of space.	123
5.16. Hand Contacts area after the <i>line</i> check. A number of internal nodes and leaf nodes are hidden for the sake of space.	124
5.17. Results of the <i>appshape</i> check, applied to different glyphs.	129
5.18. Side encoding for a glyph which features a <i>SQUARE</i> base shape.	131
5.19. Appendix encoding: <i>oncorner</i> evaluation.	133
5.20. Appendix encoding: <i>inprojection</i> evaluation.	133
5.21. Appendix encoding: <i>stangle</i> evaluation.	134
5.22. Appendix encoding: summary of all evaluations.	134
5.23. A set of glyphs belonging to the Hand Configurations area. Such symbols can be subject to drawing inaccuracy: users may draw the bottom-right appendix of each glyph either on the corner or on the side.	135
5.24. Detail of the OGR to ISWA-BIANCHINI mapping table for the Facial Expressions: Mouth area. Each ISWA-BIANCHINI code (on the left) is mapped to an OGR code (on the right).	136
6.1. High-level activity diagram describing the recognition procedure implemented by the SW-OGR Engine.	140
6.2. Package diagram of the SW-OGR Engine.	142
6.3. Class diagram of the beans package.	145
6.4. Class diagram of the <i>util</i> package: image utility library.	148
6.5. Class diagram of the <i>util</i> package: geometric utility library.	150
6.6. Class diagram of the <i>util</i> package: math utility library and comparison library.	151
6.7. Class diagram of the <i>util</i> package: shape classes.	152
6.8. Class diagram of the <i>model</i> package.	154
6.9. Class diagram of the <i>recog</i> package.	156
6.10. Fixed thresholding ($t = 100$) applied to different slices.	160
6.11. Multiple thresholding methods applied on the same slice: test slice 1.	162

6.12. Multiple thresholding methods applied on the same slice: test slice 2.	163
6.13. Test series (ground truth GT1): outcome of ME and RAE metrics for image thresholding evaluation.	165
6.14. Test series (ground truth GT2): outcome of ME and RAE metrics for image thresholding evaluation.	167
6.15. Final performance results for the thresholding evaluation test.	169
6.16. A number of circles, extracted from both DS-PEAR and DS-POITIERS datasets.	171
6.17. Application of the generalized Hough transform for the detection of curves to images which contain circles.	171
6.18. Application of the generalized Hough transform for the detection of curves to images which do not contain circles.	172
6.19. Illustration of the circle detection reliability check implemented by the SW-OGR Engine.	174
6.20. Polygon detection algorithm implemented within the SW-OGR Engine, applied to glyphs featuring rectangular shapes. The algorithm uses the Shi and Tomasi method to locate the corners in each image (top). Afterwards, the corners are processed in order to build the polygon, if this is possible (bottom).	177
6.21. Two examples of signs containing glyphs belonging to the Shoulder Movements (ShM) area. The glyphs are visible right below the head circle.	178
6.22. OGR area inference check for facial expressions. A frag is inferred to represent a left eye according to the topological relationship between its MBR and the center of the head circle.	181
6.23. OGR area inference check for facial expressions. A frag is inferred to represent a left eye according to the topological relationship between its MBR and the MBR of the head circle.	182
6.24. A set of glyphs belonging to the Head Circle and Contacts area.	184
6.25. Feature recognition applied to a set of glyphs belonging to the Head Circle and Contacts area.	185

- 6.26. A set of glyphs belonging to the Hands Configurations area. The detected base shape is marked over each glyph. 186
- 6.27. A set of glyphs belonging to the Hands Configurations area, as they appear after the thinning process. The detected base shape is marked over each glyph, and the original foreground area is visible in transparency behind each glyph. 186
- 6.28. Start point detection of the outer appendices, applied to a number of glyphs belonging to the Hands Configurations area. The detected base shape is marked over each glyph, the scan lines are visible around the base shape, and the detected foreground points along the scan lines are marked by dots. 188
- 6.29. Examples of end points in a line. 188
- 6.30. Examples of branch points in a line. 189
- 6.31. Examples of cross points in a line. 189
- 6.32. Illustration of the initial outer appendix scan applied to a number of glyphs belonging to the Hands Configurations area. Each column of the image shows: the image of the glyph, along with its detected base shape; the POIs detected following the appendices; the detected appendices, each one highlighted with a random color. 190
- 6.33. Illustration of the third appendix optimization step, performed during the recognition of glyphs belonging to the Hands configuration area. 191
- 6.34. Illustration of the fourth appendix optimization step, performed during the recognition of glyphs belonging to the Hands configuration area. 192
- 6.35. Illustration of the fourth appendix optimization step, performed during the recognition of glyphs belonging to the Hands configuration area. 193
- 6.36. Illustration of the fourth appendix optimization step, performed during the recognition of glyphs belonging to the Hands configuration area. 193
- 6.37. Screenshot representing the completed recognition of a glyph belonging to the Hands Configurations area. 195
- 6.38. Screenshot representing the completed recognition of a glyph belonging to the Hands Configurations area. 195

- 7.1. A handwritten page belonging to the DS-TEST-POITIERS dataset, as it appears before (left) and after (right) the recognition of the glyphs belonging to the Head Circles and Contacts area. 201
- 7.2. A set of glyph belonging to the Facial Expressions: Eyes area. 203
- 7.3. A handwritten page belonging to the DS-TEST-POITIERS dataset, as it appears before (left) and after (right) the recognition of the glyphs belonging to the Facial Expression: Eyes area. 204
- 7.4. Two datasets generated to test the recognition algorithms for the Hand Configurations area. The page on the left represents a number of possible variations of a hand configuration. The page on the right represents a number of printed hand configurations. 205
- 7.5. A handwritten page belonging to the DS-TEST-POITIERS dataset, as it appears before (left) and after (right) the recognition of the glyphs belonging to the Hand Configurations area. 207
- 7.6. A handwritten page belonging to the DS-TEST-PEAR dataset, as it appears before (left) and after (right) the recognition. 208
- 7.7. A handwritten page belonging to the DS-TEST-POITIERS dataset, as it appears before (left) and after (right) the recognition. 209

List of Tables

2.1. Major changes to SignWriting since 1974 to present. Information and images drawn from (Sutton, 1996).	35
2.2. Communities using SignWriting around the world, divided by continent. Information extracted from (Sutton, 1996).	36
2.3. Categories identified within the ISWA-2010. Each category is illustrated with one of its most representative glyphs. Information and images drawn from (Sutton, 1996) and (Slevinski, S.E., Jr, 2010a).	41
2.4. Group organization of Category 1: Hands, within ISWA-2010. Each group is illustrated with one of its most representative glyphs, and the relative hand configuration. Information and images drawn from (Sutton, 1996).	43
2.5. Glyph groups in ISWA-2010.	44
3.1. List of 9 tasks requested to the participants during the Think by Signs Test.	70
3.2. List of questions included in the QUIIS employed for the test.	72
3.3. Event occurrence percentage (out of the number of total events recorded) for each of the 9 tasks of the Think by Signs Test.	75

- 3.4. Percentage of occurrence of event sets for each task. The percentage of an event set for a given task is calculated out of the total events occurring during that given task. The most recurring event set is highlighted for each row. N/A is found only if no events were detected during a task. 76
- 3.5. 5 most recurring events during SWift usability test. 76
- 3.6. Satisfaction level (μ and σ) for each section of the QUIS employed for the test. 77
- 5.1. Areas defined within the OGR coding system. For each area, the table reports the code, the ISWA category or group associated to the area, and the checks that SW-OGR performs, in order to check if a glyph belongs to the area. 118
- 5.2. Hand Contacts area - *base shape* check details. For each shape, the table reports the name, the OGR code and an example image. 121
- 5.3. Hand Configurations area - *base shape* check details. For each shape, the table reports the name, the OGR code, an image of the shape, and a number of example glyphs. 126
- 5.4. Hand Configurations area - *orientation* check details. For each orientation, the table reports the name, the OGR code and a number of example images. 126
- 5.5. Hand Configurations area - *attachment* check details. For each case, the table reports the name, the OGR code and a number of example images. 127
- 5.6. Hand Configurations area - *aploc* check details. For each case, the table reports the name, the OGR code and an example image. 128
- 5.7. Appendix shapes within the Hand Configurations area. For each shape, the table reports the name, a short description and a number of example images. 128
- 6.1. Test series (ground truth GT1): misclassification error. Minimum (i.e. optimal) values for each slice are highlighted. 166
- 6.2. Test series (ground truth GT1): relative foreground area error. Minimum (i.e. optimal) values for each slice are highlighted. 167

6.3. Test series (ground truth GT2): misclassification error. Minimum (i.e. optimal) values for each slice are highlighted.	168
6.4. Test series (ground truth GT2): relative foreground area error. Minimum (i.e. optimal) values for each slice are highlighted.	169
6.5. Ratio of the diameter of head circles to the width of their slices.	180
6.6. Development Status, updated to the 27th of August 2014.	197
7.1. SW-OGR performance test: Head Circle and Contacts.	201
7.2. SW-OGR performance test: Facial Expressions: Eyes.	204
7.3. SW-OGR performance test: Hand Configurations.	206

Abstract

Abstract en Français

Les recherches et les logiciels présentés dans cette étude s'adressent à une importante minorité au sein de notre société, à savoir la communauté des sourds. Selon l'Organisation Mondiale de la Santé, cette minorité compte plus de 278 millions de personnes dans le monde. De nombreuses recherches démontrent que les sourds ont de grosses difficultés avec la langue vocale (LV – Anglais, Chinois, etc.) ce qui explique que la plupart d'entre eux préfèrent communiquer en Langue des Signes (LS). Du point de vue des sciences de l'information, les LS constituent un groupe de minorités linguistiques peu représentées dans l'univers du numérique. Et, de fait, les sourds sont les sujets les plus touchés par la fracture numérique.

Cette étude veut donc être une contribution pour tenter de combler ce fracture numérique qui pénalise les sourds. Pour ce faire, nous nous sommes principalement concentrés sur l'informatisation de SignWriting, qui constitue l'un des systèmes les plus prometteurs pour écrire la LS. Concrètement, SignWriting est un système d'écriture qui utilise des symboles pour représenter les configurations des mains, les mouvements et les mimiques du visage de la LS.

Nos travaux visent donc à projeter et élaborer un système pour développer la production et l'utilisation des ressources en LS écrites avec SignWriting. Ce système a été baptisé: SignWriting-oriented resources for the deaf (SWord). Le but final de SWord est de rendre SignWriting effectivement exploitable par les sourds aussi bien en tant que moyen de communication qu'en tant que support d'apprentissage notamment dans le domaine du numérique. SWord, un éditeur numérique permet-

tant la création de ressources numériques en LS écrites avec SignWriting, a été le premier logiciel à être inclus dans SWord. Dans la présente étude, nous souhaitons illustrer une série de fonctions mises à jour de SWift, comme par exemple la possibilité de composer des histoires entières en LS. En outre, pour évaluer la fiabilité et la facilité d'utilisation de l'application, nous avons organisé une session de tests sur un échantillon de ses principaux usagers, à savoir des personnes qui connaissent et utilisent SignWriting. Etant donné que cet échantillon est composé en majeure partie de sourds, nous avons adapté l'approche des tests de façon à ce qu'elle soit valable aussi bien pour les utilisateurs de la LS que ceux de la LV. Pour la réalisation du test, nous avons adapté une méthode classique d'évaluation de la facilité d'utilisation, à savoir le Think-Aloud Protocol (TAP) ainsi qu'un questionnaire de satisfaction très répandu, à savoir le Questionnaire for User Interaction Satisfaction (QUIS).

En dépit des efforts accomplis, nous avons constaté que les éditeurs numériques pour SignWriting, comme par exemple SWift (et de nombreux autres) sont encore bien loin d'offrir à l'utilisateur une interface en mesure d'imiter la simplicité de l'écriture manuscrite. C'est la raison pour laquelle nous avons approfondi nos recherches afin de projeter une nouvelle génération d'éditeur pour SignWriting, en mesure d'épargner à l'utilisateur d'éventuelles obligations liées au click, au glissé-déposé, à la recherche et à la navigation sur l'Interface Utilisateur (IU) pendant le processus de composition d'un signe. Notre objectif est de réaliser un mode d'interaction se rapprochant le plus possible de la méthode papier-stylo dont se servent habituellement les êtres humains pour écrire ou dessiner.

Cet objectif représente le noyau de notre étude. Pour pouvoir réaliser cette nouvelle génération d'éditeur, nous avons conçu et élaboré un logiciel préposé à la conversion électronique d'images scannées contenant des symboles manuscrits ou imprimés en textes numériques dans SignWriting. Cette technique est dénommée SignWriting Optical Glyph Recognition (SW-OGR) dans la présente étude. Apparemment SW-OGR a beaucoup de points en commun avec les techniques de Optical Character Recognition (OCR) pour LV, qui sont largement entérinées par la littérature sur la vision par ordinateur. Par conséquent, après avoir recueilli un bon nombre de textes manuscrits dans SignWriting, en fonction de leur caractéristique picturale, nous avons étudié les similitudes possibles entre le SW-OGR et les techniques à la pointe de l'OCR

pour les caractères manuscrits de la langue arabe, indienne et chinoise. Même si ces techniques adoptent des approches similaires, fondées sur des techniques de reconnaissance des formes et de l'Apprentissage Automatique, leur application dépend souvent de la notation spécifique qui doit être reconnue.

Dans le cas présent, nous n'avons pas pu suivre le même parcours pour la conception du moteur SW-OGR et nous avons dû élaborer nos propres procédures de reconnaissance. Un choix qui a été dicté par les caractéristiques de SignWriting. Primo parce que, contrairement à de nombreux alphabets, SignWriting offre une série de symboles de l'ordre de dizaines de milliers. Secundo parce que, habituellement, l'écriture à la main étant relativement imprécise, chaque symbole peut être dessiné de plusieurs façons différentes (généralement de l'ordre de dizaines de façons différentes), comme n'importe quelle notation manuscrite.

Par conséquent, il est évident que n'importe quelle formation pour une approche basée sur l'apprentissage automatique prendrait un temps exagéré et, surtout, demanderait un nombre considérable de modèles. Il y a également une autre raison qui complique l'application des techniques de reconnaissance des formes: l'absence de règles fixant le nombre, le type et la position des symboles dans un signe écrit avec SignWriting. En effet, la liberté de composition, qui est l'une des caractéristiques faisant de SignWriting un instrument très utile pour l'écriture au quotidien en LS, complique beaucoup la modélisation de la notation.

Pour toutes les raisons mentionnées ci-dessus, nous avons donc projeté et mis en place SW-OGR de façon à effectuer la reconnaissance des textes dans SignWriting en travaillant exclusivement sur les caractéristiques géométriques et topologiques des symboles et leurs relations topologiques. Nous nous sommes également appuyés sur des informations contextuelles, comme par exemple la connaissance de l'organisation de l'alphabet de SignWriting.

Dans la présente étude nous présentons également nos recherches sur l'alphabet de SignWriting visant à identifier les critères géométriques permettant de classer ses symboles, la conception de la procédure de reconnaissance, l'élaboration et la mise à l'essai du moteur SW-OGR. La première partie de la procédure vise à repérer les formes bases au sein des fragments présents dans le texte écrit (cercles, polygones, etc.). C'est la seule étape au cours de laquelle nous pouvons exploiter les procédures d'apprentissage automatique existantes. Les formes détectées et leurs caractéristiques sont utilisées pour inférer la région du corps et

le type d'élément non-anatomique (mouvement, contact, etc.) que le fragment peut décrire.

A partir de là, les critères identifiés correspondent à une série de vérifications effectuées pour reconnaître les caractéristiques du symbole. Les résultats de ces vérifications sont ensuite utilisés pour construire un code, qui est lui-même utilisé pour identifier le symbole.

Le moteur SW-OGR a vocation à accomplir une double fonction: en premier lieu, il peut être intégré dans un éditeur de SignWriting existant, tel que SWift, afin de fournir un support immédiat à l'écriture manuscrite et de rendre le processus de composition beaucoup plus rapide et pratique pour un usage quotidien. A signaler que depuis le début de nos travaux, nous étions conscients de la présence (et des dimensions importantes) d'un certain nombre de corpus manuscrits en SignWriting recueillis par diverses communautés dans le monde. Cette documentation est un bien précieux qui pourrait être beaucoup plus utile encore si elle était numérisée. En second lieu, SW-OGR est donc en mesure de numériser ces corpus de façon « intelligente », à savoir qu'il ne se limite pas à effectuer un simple « balayage » des documents mais, à travers la reconnaissance, il est en mesure de recueillir toutes les informations sur le rapport entre les signes et les symboles qui les composent, et permet ainsi à la communauté scientifique de réaliser tout type d'analyse linguistique sur des séries de données complètes en SignWriting.

Abstract in English

The studies and the software presented in this work are addressed to a relevant minority of our society, namely deaf people. According to the World Health Organization, such “minority” currently counts more than 278 million people worldwide. Many studies demonstrate that, for several reasons, deaf people experience significant difficulties in exploiting a Vocal Language (VL - English, Chinese, etc.). In fact, many of them prefer to communicate using Sign Language (SL). As computer scientists, we observed that SLs are currently a set of underrepresented linguistic minorities in the digital world. As a matter of fact, deaf people are among those individuals which are mostly affected by the digital divide.

This work is our contribution towards leveling the digital divide affecting deaf people. In particular, we focused on the computer handling of SignWriting, which is one of the most promising systems devised to write SLs. More specifically, SignWriting is a writing system which uses visual symbols to represent the handshapes, movements, and facial expressions of SLs.

Our efforts aim at designing and implementing a framework to support the production and the use of SignWriting-oriented resources for the deaf (SWord). The ultimate purpose of SWord is to make SignWriting effectively exploitable as a communication mean and a learning support for deaf people, especially in the digital world. The first software to be included into the SWord framework was SWift, a digital editor which allows the creation of digital SL resources written in SignWriting. In this work, we present a number of upgraded features of SWift, such as the possibility to write signed stories. Moreover, to assess the reliability and the usability of such application, we conducted a test session with its main target users, i.e. people which are proficient in SignWriting. Since our sample was mainly composed by deaf people, we adapted our testing approach to be viable for both SL and VL users. In order to perform the assessment, we adapted a popular usability testing methodology, namely the Think-Aloud Protocol (TAP), and a widespread customizable questionnaire, namely the Questionnaire for User Interaction Satisfaction (QUIS).

Despite our efforts, we observed that SignWriting digital editors, such as SWift (and many others), are still far from granting the user an interface which is able to emulate the simplicity of handwriting. For this

reason, we continued working towards the possibility to design a new generation of SignWriting editors, able to lift the user of any burden related to clicking, dragging, searching, browsing on the User Interface (UI) during the composition process of a sign. Our aim is to implement an interaction style which is as similar as possible to the paper-pencil approach that humans normally use when writing or drawing.

The main part of the present work deals with this latter goal. To make such new generation of editors feasible, we designed and implemented a software application whose purpose is to operate the electronic conversion of scanned images containing handwritten or printed SignWriting symbols into machine-encoded SignWriting texts. Such technique is referred to as the SignWriting Optical Glyph Recognition (SW-OGR) in the present work.

Apparently, the SW-OGR topic shares much in common with the Optical Character Recognition (OCR) techniques for VLs, which are well consolidated within the computer vision literature. More specifically, after gathering a fair number of handwritten SignWriting texts, and given their pictorial nature, we investigated the possible similarities between the SW-OGR and the modern OCR techniques employed to perform the recognition of Arabic, Indian and Chinese handwritten characters. Even if such techniques adopt similar approaches, based on pattern recognition and machine learning, their implementation may often depend on the specific notation to be recognized.

In our case, we could not follow a similar line for the design of the SW-OGR Engine, and we were forced to devise our own recognition procedures. The reason for this choice is the very particular nature of SignWriting. First of all, unlike most alphabets, SignWriting features a number of symbols which is in the order of the tens of thousands. Moreover, since handwriting is usually quite inaccurate, each symbol can be drawn in a number (usually in the order of tens) of different ways, as in any handwritten notation. As a consequence, it is evident that any training to enable a machine learning approach would require an unreasonable amount of time and, most of all, of training templates. A further reason makes the application of pattern recognition techniques very difficult, i.e. the total lack of rules regulating the number, type and position of the symbols within a SignWriting sign. In fact, the composition freedom, which is one of the features which make SignWriting a very handy tool for everyday SL writing, also makes the modeling of the notation very difficult.

For the above reasons, we designed and implemented SW-OGR to perform the recognition of SignWriting texts by only working with the geometric and topological features of the symbols, and with their topological relationships. We also relied on context-dependent information, such as the knowledge of the organization of the SignWriting alphabet. In this work we present the studies performed on the SignWriting alphabet in order to identify geometric criteria to classify its symbols, the design of the recognition procedure, and the development and testing of the SW-OGR engine. The starting points of the procedure aims at identifying base shapes within the fragments of the written text (circles, polygons, etc.). This is the only step where we can exploit existing machine learning procedures. Detected shapes and their features are used to infer the body area or kind of non-anatomical element (movement, contact, etc.) that the fragment may depict. From this point on, the identified criteria correspond to a series of checks performed to recognize the features of the symbol. The result of the checks are used to build a code, which is finally used to identify the symbol.

The engine is intended to serve a twofold purpose: first of all, it can be embedded within existing SignWriting editors, such as SWift, in order to provide a prompt support for handwriting, and make the composition process much faster and comfortable for everyday use. In addition, since the beginning of our work, we were aware of the presence (and of the considerable size) of a number of handwritten SignWriting corpora gathered from different communities around the world. Those corpora are an invaluable asset, and they could become even more useful if digitalized. SW-OGR is able to digitalize such corpora in a smart way, since it does not simply perform a “scan” of the documents, but, through the recognition, it is able to gather any information about the relationship between a sign and the symbols that compose it, thus allowing the research community to perform any kind of linguistic analysis on whole SignWriting datasets.

Abstract in Italiano

Gli studi ed i software presentati in questo lavoro sono indirizzati ad una importante minoranza nella nostra società, ossia quella costituita dalle persone sorde. Attualmente, secondo la World Health Organization, tale “minoranza” è costituita da più di 278 milioni di persone in tutto il mondo. Molti studi dimostrano che, per diverse ragioni, le persone sorde incontrano serie difficoltà nell'utilizzo di una Lingua Vocale (LV - Inglese, Cinese, ecc.). Infatti, molti di loro preferiscono comunicare usando la Lingua dei Segni (LS). Da un punto di vista delle scienze dell'informazione, si può osservare che le LS costituiscono attualmente un gruppo di minoranze linguistiche poco rappresentate nel mondo digitale. Di fatto, le persone sorde sono tra gli individui maggiormente affetti dalla digital divide.

Il presente lavoro rappresenta il nostro contributo per ridurre la digital divide che colpisce le persone sorde. In particolare, ci siamo concentrati sull'informatizzazione di SignWriting, che è uno dei sistemi più promettenti ideati per scrivere la LS. Più specificamente, SignWriting è un sistema di scrittura che utilizza simboli per rappresentare le configurazioni delle mani, i movimenti e le espressioni facciali della LS.

I nostri sforzi mirano a progettare e implementare un framework per favorire la produzione e l'utilizzo di risorse in LS scritte con SignWriting, il nome del framework è SignWriting-oriented resources for the deaf (SWord). Il fine ultimo di SWord è di rendere SignWriting effettivamente sfruttabile come mezzo di comunicazione e come supporto per l'apprendimento per le persone sorde, soprattutto nel mondo digitale. Il primo software ad essere incluso in SWord è stato SWift, un editor digitale che permette la creazione di risorse digitali in LS scritte con SignWriting. In questo lavoro, presentiamo una serie di features aggiornate di SWift, come ad esempio la possibilità di comporre intere storie in LS. Inoltre, per valutare l'affidabilità e l'usabilità di tale applicazione, abbiamo condotto una sessione di test con un campione dei suoi principali utenti finali, ossia persone che conoscono ed usano SignWriting. Dato che il nostro campione è composto principalmente da persone sorde, abbiamo adattato il nostro approccio di test in modo da essere valido sia per gli utenti di LS che di LV. Al fine di eseguire il test, abbiamo adattato un popolare metodo per la valutazione dell'usabilità, ossia il Think Aloud Protocol (TAP), e un questionario personalizzabile molto diffuso, vale a dire il Questionario per la User Interaction Satisfaction

(QUIS).

Nonostante i nostri sforzi, abbiamo osservato che gli editor digitali per SignWriting, come ad esempio SWift (e molti altri), sono ancora molto lontani dal fornire all'utente un'interfaccia che sia in grado di emulare la semplicità della scrittura a mano. Per questo motivo, abbiamo continuato a lavorare verso la possibilità di progettare una nuova generazione di editor per SignWriting, in grado di sollevare l'utente da eventuali oneri legati a click, drag-and-drop, ricerca e navigazione sull'Interfaccia Utente (UI) durante il processo di composizione di un segno. Il nostro obiettivo è realizzare uno stile di interazione che sia il più simile possibile al metodo carta-e-penna che gli esseri umani normalmente utilizzano durante la scrittura o il disegno.

La parte principale del presente lavoro affronta quest'ultimo obiettivo. Per rendere realizzabile questa nuova generazione di editor, abbiamo progettato e implementato un software il cui scopo è quello di operare la conversione elettronica di immagini scansionate contenenti simboli scritti a mano o stampati in testi digitali in SignWriting. Nel presente lavoro, tale tecnica viene definita come SignWriting Optical Glyph Recognition (SW-OGR).

Apparentemente, SW-OGR ha molto in comune con le tecniche di riconoscimento ottico dei caratteri (OCR) per LV, che sono ben consolidate nella letteratura relativa alla computer vision. In particolare, dopo aver raccolto un discreto numero di testi scritti a mano in SignWriting, e data la loro natura pittorica, abbiamo studiato le possibili analogie tra la SW-OGR e le moderne tecniche di OCR per i caratteri scritti della lingua araba, indiana e cinese. Anche se tali tecniche adottano approcci simili, basati su tecniche di pattern recognition e machine learning, la loro attuazione può spesso dipendere dalla specifica notazione che deve essere riconosciuta.

Nel nostro caso, non abbiamo potuto seguire una linea simile per la progettazione del motore SW-OGR, e abbiamo elaborato le nostre procedure di riconoscimento. La ragione di questa scelta è la particolare natura di SignWriting. Prima di tutto, a differenza di molti alfabeti, SignWriting presenta una serie di simboli che è dell'ordine delle decine di migliaia. Inoltre, poiché la scrittura a mano è di solito piuttosto imprecisa, ogni simbolo può essere disegnato in un numero (di solito dell'ordine di decine) di modi diversi, come in qualsiasi notazione manoscritta. Di conseguenza, è evidente che qualsiasi training per consentire un approccio basato sul machine learning richiederebbe una quantità ir-

ragionevole di tempo e, soprattutto, di templates. Un ulteriore motivo rende l'applicazione di tecniche di pattern recognition molto difficile, cioè la totale mancanza di norme che regolino il numero, il tipo e la posizione dei simboli all'interno di un segno scritto con SignWriting. Infatti, la libertà composizione, che è una delle caratteristiche che rendono SignWriting uno strumento adatto per l'utilizzo quotidiano, rende anche la modellizzazione della notazione molto difficile.

Per le ragioni di cui sopra, abbiamo progettato e implementato SW-OGR in modo da eseguire il riconoscimento di testi in SignWriting lavorando esclusivamente con le caratteristiche geometriche e topologiche dei simboli, e con le loro relazioni topologiche. Abbiamo anche fatto affidamento su informazioni contestuali, come ad esempio la conoscenza dell'organizzazione dell'alfabeto di SignWriting.

In questo lavoro presentiamo i nostri studi sull'alfabeto di SignWriting al fine di identificare i criteri geometrici per classificare i suoi simboli, il design della procedura di riconoscimento, e lo sviluppo e il testing del motore SW-OGR. La parte iniziale della procedura mira ad individuare le forme base all'interno dei frammenti presenti del testo scritto (cerchi, poligoni, ecc). Questo è l'unico passo in cui possiamo sfruttare procedure di machine learning esistenti. Le forme rilevate e le loro caratteristiche sono usate per inferire la regione del corpo o il tipo di elemento non-anatomico (movimento, contatto, ecc) che il frammento può descrivere. Da questo punto in poi, i criteri individuati corrispondono a una serie di controlli effettuati per riconoscere le caratteristiche del simbolo. Il risultato dei controlli vengono utilizzati per costruire un codice, che viene infine utilizzato per identificare il simbolo.

Il motore SW-OGR è destinato a servire un duplice scopo: in primo luogo, esso può essere integrato all'interno di editor di SignWriting esistenti, come SWift, al fine di fornire un supporto rapido per la scrittura a mano, e rendere il processo di composizione molto più veloce e comodo per l'uso quotidiano. Inoltre, fin dall'inizio del nostro lavoro, eravamo consapevoli della presenza (e della notevole dimensione) di un certo numero di corpus manoscritti in SignWriting, raccolti da diverse comunità in tutto il mondo. Questi corpus sono un bene prezioso, e potrebbero diventare ancora più utili se digitalizzati. SW-OGR è in grado di digitalizzare tali corpus in modo intelligente, in quanto non si limita a eseguire una semplice "scansione" dei documenti, ma, attraverso il riconoscimento, è in grado di raccogliere tutte le informazioni sul rapporto tra i segni e i simboli che li compongono, permettendo così

alla comunità scientifica di eseguire qualsiasi tipo di analisi linguistica su interi dataset in SignWriting.

Introduction

The subject of the present work lies in the middle between image processing and human-computer interaction (HCI). The title reads: *Informatisation of a graphic form of Sign Languages - application to SignWriting*, and it is important to spend a few words to clarify its meaning. First of all, Sign Language (SL) is the visual-gestural language used by many deaf people to communicate with each other. SL is deeply different from the Vocal Language (VL) used by hearing people, such as English, Chinese, etc. This is mainly due to the fact that the cognitive structures underlying the language processing of deaf people are deeply different from those exploited by hearing people. In fact, such structures reflect the highly visual perception experienced by people who mainly feel the world surrounding them through their eyes.

Like many other languages in the world, SLs do not currently have a widely acknowledged *graphic* (or *written*) *form*. The main difference between the two situations is that in the former case the problem is mostly limited to small and isolated communities, while SLs, which are different in the different countries like VLs, are used by a community which shares its communication space with another (hearing) community, whose VL is the dominating one. As a matter of fact, the deaf communities around the world do not currently share a common writing system to represent their preferred language. This still happens notwithstanding the fact that a number of writing systems have been developed for SLs, by different research teams and for different purposes (see Chapter 1). Actually, choosing one writing system for SLs is subordinated to the goal that one must pursue. In our case, given our goal to increase the accessibility of electronic resources to deaf people in every day life, SignWriting proved the most appropriate candidate to

work with, since it has features that can rarely be found in other SL writing systems. More specifically, SignWriting is a writing system which uses visual symbols to represent the handshapes, movements, and facial expressions of SLs. Its high iconicity, and the possibility to be employed in everyday use made it the ideal candidate for a wide diffusion, and for this work (see Chapter 2).

Unfortunately, neither SLs, and even less any of the systems devised to write them, are adequately, extensively and ubiquitously exploited to transmit information in the digital world. Most digital artifacts, in fact, are available only in VL, whether they are applications, content, websites, etc. Little attention is paid to accessibility design for deaf people, which is often carried out using solutions on the edge of the workaround, i.e. through textual captioning and audio-content transcription. As detailed in Chapter 1, this kind of design may support the needs of people who become deaf after the acquisition of speech and language (post-lingual deafness), but issues related to pre-lingual deafness are seldom and poorly addressed. Actually, a deaf-oriented accessibility design based on SL support is still far from being realized. Videos, more specifically *signed videos*, are typically the most widespread technique for SL inclusion. However, even if many deaf communities have recently replaced many functions of *writing* by using signed videos, this is not always possible or appropriate. As an example, it is not possible to take notes with a video, or to annotate a web resource (tagging) or to enter a query on a search engine. In other words, videos lack the ease of handling and the variegated usability of a written expression.

The belief underlying the present work is that the informatisation of SLs is of paramount importance to achieve an effective deaf-oriented accessibility design, and to ultimately mitigate the impact of the digital divide on deaf people. First of all, in order to achieve this goal, it was necessary to design a framework to support the production and the use of SignWriting-oriented resources for the deaf (SWord). SWord is a framework designed to include software for the acquisition, management and dissemination of signs and signed stories written in SignWriting. The ultimate purpose of SWord is to make SignWriting effectively exploitable as a communication mean and a learning support for deaf people, especially in the digital world (see Chapter 3). The first software to be included into the SWord framework was SSwift, a digital editor which allows the creation of digital SL resources written in SignWriting. In this work, we present a number of upgraded features of SSwift, such as

the possibility to write signed stories. Moreover, we conducted a test session to evaluate the usability and the correct functioning of SWift; in order to do this, a sample of the main target users of SignWriting digital editors was gathered. Since most participants were deaf, the tools and the methodologies chosen for the test were adapted to work with deaf people too (see Chapter 3.2)

The HCI, however, is not the core topic of this dissertation. In fact, despite our efforts, we observed that handwriting signs is still much easier than using any SignWriting digital editor, such as SWift, to compose them. For this reason, we designed and implemented a software methodology whose purpose is to operate the electronic conversion of scanned images containing handwritten or printed SignWriting symbols into machine-encoded SignWriting texts. Such technique is referred to as the SignWriting Optical Glyph Recognition (SW-OGR) in the present work (see Chapter 4, 5, 6 and 7).

Due to the huge number of symbols to recognize (about 40.000), it was not possible to rely on Machine Learning techniques, since any training procedure aiming at recognizing the whole set of glyphs would have been unsustainably expensive. For this reason, SW-OGR performs the recognition of SignWriting texts by only working with processing rules related to the geometric and topological features of the symbols, and with their topological relationships. We also relied on context-dependent information, such as the knowledge of the organization of the SignWriting alphabet. In this work we present the studies performed on the SignWriting alphabet in order to identify geometric criteria and rules to classify its symbols, the design of the recognition procedure, and the development and testing of the SW-OGR engine. The engine is intended to serve a twofold purpose: first of all, it can be embedded within existing SignWriting editors, such as SWift, in order to provide a prompt support for handwriting, and make the composition process much faster and comfortable for everyday use. In addition, SW-OGR is able to digitize paper SignWriting corpora in a smart way, since it does not simply perform a “scan” of the documents, but, through the recognition, it is able to gather any information about the relationship between a sign and the symbols that compose it, thus allowing the research community to perform any kind of linguistic analysis on newly-acquired SignWriting datasets.

The present dissertation is composed by 7 chapters. Chapter 1 covers the necessary background about deafness and language; Chapter 2 pro-

vides a detailed description of SignWriting; Chapter 3 introduces the SWord framework and SWift. The remaining chapters cover SW-OGR. Chapter 4 introduces SW-OGR and the related work about Optical Character Recognition (OCR); Chapter 5 covers the studies on the SignWriting alphabet in order to identify the features to be exploited during the recognition procedure; Chapter 6 details the design and the implementation of SW-OGR; finally, Chapter 7 covers the testing of the application.