HamNoSyS2SiGML: Translating HamNoSys Into SiGML

Carolina Neves, Luisa Coheur, Hugo Nicolau
Instituto Superior Técnico, Universidade de Lisboa/INESC-ID
Lisboa, Portugal
{name.surname}@tecnico.ulisboa.pt

Abstract

Sign Languages are visual languages and the primary means of communication used by Deaf people. However, the majority of the information available online is presented through written form. Hence, it is not of easy access to the Deaf community. Avatars have gained an increase of interest due to their potential in automatically generating signs from text. Synthetic animation of conversational agents can be achieved through the use of notation systems. HamNoSys is one of these systems, which describes movements of the body through symbols. SiGML is an XML-compliant machine-readable format that enables avatars to animate HamNoSys symbols. However, there are no freely available open-source libraries that allow the conversion from HamNoSys to SiGML. In this paper, we present our open-source and cross-platform tool that performs such conversion. This system represents a crucial intermediate step in the broader pipeline of animating signing avatars. Finally, we describe two cases studies to illustrate different applications of our tool.

Keywords: HamNoSys, SiGML, translation, signing avatar

1. Introduction

Sign languages are visual gesture languages performed through the use of hands, facial, and body expressions. According to the World Federation of the Deaf[1] there are over 300 sign languages and 70 million deaf people using them around the world. Sign languages are highly structured languages with linguistic rules distinct from their spoken counterparts without a standard written form. However, the vast majority of information available online is provided in spoken or written language, which excludes sign languages. Many communication barriers exist for sign language users (Kushalnagar, 2019), and signing avatars (computer animations of humans) have the potential to break down these barriers for Deaf people who prefer sign language or have lower literacy in written language (Kushalnagar, 2019). For instance, these avatars could automatically replace the displayed text with sign language animations.

Current pipelines typically generate signing avatars based on a symbolic representation previously prepared by a human annotator of signed content (Al-khazraji et al., 2018; Efthimiou et al., 2019; Elliott et al., 2000; Adamo-Villani and Wilbur, 2015; Zwitserlood et al., 2004); that is, avatars are built based on annotated datasets that relate gestural information with linguistic information.

While multiple notation systems have been proposed (Costa and Dimuro, 2003; Elliott et al., 2000; Hanke, 2004), there is no universal standard. Annotations are often in gloss, a form of transliteration where written words in spoken languages are used to represent signs. The Hamburg Notation System for Sign Languages (HamNoSys) (Hanke, 2004) is one of the most popular, designed to capture detailed human movements and body positioning for computer modelling. The notation system uses symbols to describe parameters of signs, such as hand shapes, hand orientations, movements, and non-manual components. Although HamNoSys symbols are readable by humans, notation systems typically require intermediary XML-based markup language representations that are computationally compatible and readable (Costa and Dimuro, 2003; Zwitserlood et al., 2004). For instance, Signing Gesture Markup Language (SiGML) (Zwitserlood et al., 2004) is compatible with HamNoSys.

Multiple sign language avatar projects have been using HamNoSys and SiGML. For instance, eSIGN's (Essential Sign Language Information on Government Networks) (Zwitserlood et al., 2004) avatar uses both notation systems to synthesize signing animations. The avatar will receive the SiGML and will then display the requested signs. More recently, SiS-Builder (Efthimiou et al., 2019) is a sign language tool that provides a signer friendly graphical user interface. Users can create SiGML scripts either by entering HamNoSys strings of already stored signs or by creating HamNoSys lemmas online, up to a limit of four lemmas. Users may also add non-manual components to the sign. Afterwards, the respective SiGML script is generated and stored. However, the previously mentioned tools are not open source.

Although signing avatars often leverage HamNoSys annotations, to best of our knowledge, there are no freely available open-source libraries that allow the conversion to SiGML, which leads to duplication of work and inconsistencies in the conversion process. Moreover, it limits the potential of HamNoSys to be extended, mainly when dealing with unforeseen hand signs and non-manual components, resulting in custom-made notation "silos" that are never shared or standardized.

In this paper, we present an open-source library that enables real-time conversion from HamNoSys to SiGML[2]. The library is language independent, deals with any number of glosses and symbols, and can be included in popular 3D development platforms (e.g. Unity). The contributions of this paper are: 1) a freely available open-source tool that converts HamNoSys to SiGML, 2) the possibility to extend

---

[1] http://wfdeaf.org/our-work/

[2] https://github.com/carolNeves/HamNoSys2SiGML

the notation and its SiGML, 3) a parser capable of converting the annotations from ELAN[3] to the correct SiGML, and 4) combine our tool with development platforms, such as Unity[4]. We further illustrate the library's capabilities with two case studies.

The paper starts with an overview of HamNoSys and its XML-notation, SiGML, followed by a description of our library. Next, we illustrate how the community can leverage it. Finally, we present the conclusions withdrawn from our work as well as future work.

2. From HamNoSys to SiGML

2.1. HamNoSys

HamNoSys is an alphabetic system describing signs mostly on a phonetic level. The following description of the system is based on (Hanke, 2004). This notation system is a combination of iconic and easily recognizable symbols which cover the parameters of hand shape, hand configuration, location and movement. HamNoSys can be internationally used since it does not rely on conventions that differ from country to country (Hanke, 2004; Kaur and Kumar, 2016).

2.1.1. Hand shapes

The description of hand shape is composed of basic hand forms and thumb combinations. These can be combined with diacritic signs for thumb positions and bending. Additional description concerning the fingers involved or the form of individual fingers can be specified. In Figure 1 we can see a simple hand shape for the Portuguese word "bolo" ("cake" in English).

2.1.2. Hand orientations

The description of the hand orientation is composed of two components: extended finger and palm orientation. The former provides information regarding the direction of an extended finger related to the signer's body (signer's view, birds' view, and view from the right). The later is relative with the former. For a given extended finger, it indicates an orientation of the palm around the shaft of the hand. A practical example is provided in Figure 1.

2.1.3. Hand locations

The locations of the hand can also be split into two components: The first provides information of the hand location in respect to other body parts, as the second determines the distance of the hand to this location. If the later is missing, a "natural" distance is assumed. In case both components are omitted, a neutral space is assumed. Such space is located in front of the upper part of the body. As shown in Figure 1, both the hand location and hand proximity are provided.

2.1.4. Movements

Movements can be distinguished between straight, curved and zigzag lines, circles and similar forms. These can either be performed sequentially or co-temporally. Also, repetitions of movements can be specified. In the case of two-handed signs, it is possible to differentiate the actions for each hand.

2.1.5. Two-handed signs

HamNoSys also supports two-handed signs. These can present symmetric or asymmetric configurations. A symmetric marker is used to specify how both hands should behave.

2.1.6. Non-manual components

Non-manual components are all those regarding body and face without considering hands, such as head, eyebrows, eyes, cheeks, mouth, torso and shoulder movements. The description of these components in HamNoSys is rather limited. For each movement, it is possible to specify an articulator to replace the hand. An articulator allows appropriate descriptions for shoulder shrugging, head movements but not necessarily facial expressions or mouth movements.

2.2. SiGML

HamNoSys provides readable symbols to the human eye, but in terms of computer processing, it is not as straightforward. ViSiCAST (Elliott et al., 2000) was the pilot project in this field. This project focused on the definition of the XML-compliant representation of HamNoSys symbols, SiGML. SiGML is a machine readable input of HamNoSys (Zwitserlood et al., 2004), which describes the gestures. This XML framework can be used to drive avatars if the signing expressed in HamNoSys is properly converted to it. Figure 1 also illustrates how the word "BOLO" is coded in SiGML (right column).
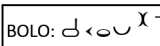

Figure 1: Representation of the Portuguese SL sign for "BOLO" (cake in English) in HamNoSys (left) and its codification in SiGML (right).

3. HamNoSys2SiGML tool

HamNoSys2SiGML is an automation system designed to receive a set of HamNoSys codes with the optional addition of its own glosses, in said order, and return a SiGML. Our system represents an intermediate step in the pipeline of synthetic animation. Even though several tools were previously developed, to the best of our knowledge, none of them is freely available. The only requirement to use our tool is to have Python 3.7.3+ installed in the user's device.

3.1. Architecture

The architecture of the system consists mainly of three major steps and three minor steps, presented in Figure 2.

The recognized input of our program can be distinguished into two scenarios: a set of HamNoSys symbols or a set

---
[3] https://tla.mpi.nl/tools/tla-tools/elan/
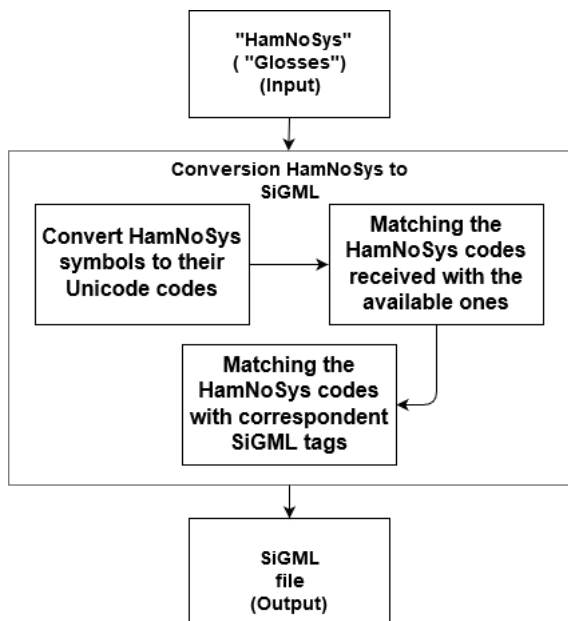[4] https://unity.com/

Figure 2: Architecture of the system.

of HamNoSys symbols and their corresponding glosses. In both scenarios, the input must be presented within quotes, as shown in Figure 3. For the second scenario, both the number of HamNoSys blocks and glosses must coincide; otherwise, the input will not be supported by the program. An example is given in Figure 3 in which the system receives two sets of HamNoSys symbols and two glosses, "Bom dia" ("Good morning" in English) in Portuguese SL (PSL). [5]

Figure 3: HamNoSys2SiGML receives the HamNoSys symbols in PSL (first input value) and two glosses "bom dia" ("good morning") (second input value).

3.2. Processing the data

The system will read the input and evaluate which scenario is applied by counting the numbers of arguments presented in quotes. If the input contains two sets of information in quotes, the program will assume it is receiving both Ham-NoSys symbols and their glosses. In contrast, if the input only contains one set of information in quotes, it will assume

---

[5] The HamNoSys symbols can not be displayed in the command line since they are not machine-readable.

it is just receiving HamNoSys symbols. The program has no limit over the maximum number of HamNoSys symbols possible to receive, ergo, neither it does over the number of glosses.

The HamNoSys symbols received initially cannot be directly read; for this reason, they must be converted into machine-readable content. These symbols are available as an Unicode font with the characters mapped into the Private Use area of Unicode. The Private Use area of Unicode is a range of code points that are intentionally left undefined so that third parties can define their characters without conflicting with already existing Unicode characters. In this case, we use HamNoSys symbols. Each symbol is associated with a code of exactly four characters. The correct association between both the symbol and its corresponding code is performed by the system in step "Convert HamNoSys symbols to their Unicode codes" (Figure 2). We provide an example in Figure 4, in which the values in the column *Symbols* are the input received by the program, and its respective code is correspondent to the values in the column *Codes*.

3.3. SiGML generation

The system has access to approximately 210 HamNoSys symbols and their corresponding SiGML tags, provided by the authors of SiS-Builder (Efthimiou et al., 2019). Once all HamNoSys codes received are converted to their respective set of Unicode characters, it is possible to match them with those accessed by the program ("Matching the Ham-NoSys codes received with the available ones", Figure 2). Afterwards, the correct SiGML tag is accessed, which corresponds to the final step "Matching the HamNoSys codes with correspondent SiGML tags" in Figure 2. In Figure 4 the SiGML tags are present in the column most to the left (*HamNoSys token*).

Once all the associations are performed, a SiGML file is written to the command line (Figure 3) designated as the last step of the architecture in Figure 2. In case only the HamNoSys symbols are provided, the output will only differ regarding the glosses information (*gloss = "BOM"*), which will be non-existent.

HamNoSys token	HamNoSysUnicode	
	Codes	Symbols
hamfinger2345	E005	⠻
hampinch12	E006	◠

Figure 4: Correspondence between SiGML tags, Ham-NoSys Unicode codes and HamNoSys symbols, in the respective order.

3.4. Extending the notation

The presented tool can be further extended by the user. Taking into consideration the currently limited facial expressions codes available in HamNoSys, a promising possibility resides in the extension of the notation system for this type of content.

As previously mentioned in section 3.3., the system has access to approximately 210 HamNoSys codes and their corresponding SiGML tags, which is saved in a text file (specified in the README file). With the aim of improving the content of this document, the user simply has to add entries to this file. Each entry must have the SiGML tags created and their respective HamNoSys Unicode codes.

4. Leveraging HamNoSys2SiGML

Our tool brings value to a range of different fields, such as linguists and developers. We describe two cases studies to illustrate the possible usages of HamNoSys2SiGML. The installation of Python 3.7.3+ is required for both.

4.1. First Case Study: ELAN

ELAN[6] is one of the most popular annotation tools, distributed with free licenses, which allow their non-commercial use. Linguists widely use the tool in the process of annotating content from videos in sign languages. ELAN provides as output an annotation file which contains all the information contained in its different tiers.

For our case study, the output was exported as an HTML with two tiers (Glosses and HamNoSys). In order to read the input from this HTML, a parser was developed. In case ELAN original file has different tiers, this parser requires adjustments. While reading the data from the HTML, the conversion from HamNoSys symbols to their codes was performed. This parser returns the HamNoSys Unicode codes followed by their glosses. Nevertheless, this input is also accepted by our tool. Therefore, no changes are required to the original code. The returned SiGML for this HTML can be used to animate an avatar from CoffeeScript WebGL ARP Signing Avatars[7].

By combining ELAN's functionalities with our tool's, we provide linguists with a solution for animating an online avatar fed by their annotations.

4.2. Second Case Study: ELAN with Unity

Unity[8] is one of the most popular game engines available, packed with tools that allow for powerful, yet simple, 2D and 3D game development, with free versions unrestricted. By integrating the HTML exported from ELAN with Unity, developers can animate a state-of-the-art avatar of choice. For such purpose, the scripts should be within the Unity's project folder *"/Assets/PythonScripts/"*. It is worth mentioning that this code will only work in Windows. Firstly the python installation will be verified by Unity. Afterwards, the platform will start new processes for each script as well as for the parser and conversion tool (HamNoSys to SiGML). These processes will open Windows' command line in the background and run the scripts. The SiGML files generated, by default, will be saved to *"/Assets/SiGML_Files/"*.

This combination is advantageous for diversified teams of linguists and developers which aspire to develop a complete pipeline in the synthetic animation of an avatar.

5. Conclusions and Future Scope

Information currently available online is provided in written form, which is not of easy access to Deaf people. The ideal method for displaying information to the Deaf community would be through their sign language. An accessible approach to create content in SL is through the transcription of signs into HamNoSys, followed by the proper animation by an avatar.

Our tool allows for the translation from a set of HamNoSys symbols, and optionally their corresponding glosses, to SiGML, without the dependency of external systems.

We describe two cases studies illustrating different scenarios of use for our tool, providing an overview of the range of possibilities in which our system can be leveraged.

The described tool can be used as an essential intermediate step in the broader pipeline of producing content in SL. Further complementary work to this tool can be added to complete this pipeline. Namely the development of a connection between the SiGML produced and the avatar to animate.

6. Acknowledgements

7. Bibliographical References

Adamo-Villani, N. and Wilbur, R. B. (2015). ASL-Pro: American Sign Language Animation with Prosodic Elements. In Margherita Antona et al., editors, Universal Access in Human-Computer Interaction. Access to Interaction, pages 307–318, Cham. Springer International Publishing.

Al-khazraji, S., Berke, L., Kafle, S., Yeung, P., and Huenerfauth, M. (2018). Modeling the Speed and Timing of American Sign Language to Generate Realistic Animations. In Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS '18, pages 259–270, New York, NY, USA. ACM.

Costa, A. C. d. R. and Dimuro, G. P. (2003). SignWriting and SWML: Paving the way to sign language processing. *Traitement Automatique des Langues de Signes, Workshop on Minority Languages*, pages 11–14.

Efthimiou, E., Fotinea, S.-E., Goulas, T., Vacalopoulou, A., Vasilaki, K., and Dimou, A.-L. (2019). Sign Language Technologies and the Critical Role of SL Resources in View of Future Internet Accessibility Services. 7(1):18.

Elliott, R., Glauert, J. R., Kennaway, J., and Marshall, I. (2000). The development of language processing support for the visicast project. In ASSETS, volume 2000, page 4th.

Hanke, T. (2004). HamNoSys-Representing sign language data in language resources and language processing contexts. *Proceedings of the Workshop on Representation*

---

[6] https://tla.mpi.nl/tools/tla-tools/elan/
[7] http://vhg.cmp.uea.ac.uk/tech/jas/vhg2019/cwa/TwoAvServer.html
[8] https://unity.com/

and Processing of Sign Language, Workshop to the forth International Conference on Language Resources and Evaluation (LREC'04), pages 1–6.

Kaur, K. and Kumar, P. (2016). HamNoSys to SiGML Conversion System for Sign Language Automation. *Procedia - Procedia Computer Science*, 89:794–803.

Kushalnagar, R. (2019). Deafness and Hearing Loss. In Yeliz Yesilada and Simon Harper (Eds.), *Web Accessibility: A Foundation for Research*. London:Springer London, pp. 35–47.

Zwitserlood, I., Verlinden, M., Ros, J., and Van Der Schoot, S. (2004). Synthetic signing for the deaf: Esign. In Proceedings of the conference and workshop on assistive technologies for vision and hearing impairment, CVHI, Granada, Spain. Citeseer.