Sign Language Translation Mobile Application and Open Communications Framework

Deliverable 5.5: Second Sign language-specific lexicon and structure

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101017255

Project Information
Project Number: 101017255
Project Title: SignON: Sign Language Translation Mobile Application and Open Communications Framework
Funding Scheme: H2020 ICT-57-2020
Project Start Date: January 1st 2021

Deliverable Information
Title: Second Sign language-specific lexicon and structure
Work Package: WP5 - Target Message Synthesis
Lead beneficiary: UPF / GTI
Due Date: 28/02/2023
Revision Number: V0.2
Authors: Dimitar Shterionov, Irene Murtagh, Bram Vanroy, Rahul Agrahari
Dissemination Level: PU
Deliverable Type: Other

Overview: In this deliverable we describe the second sign language (SL)-specific lexicon and structure together with (i) the tools to generate lexicons for different SLs from monolingual (videos) and parallel (videos aligned with lexicon entries) data; (ii) the pipelines for synthesis of SL avatars. Furthermore, we provide an overview of the differences between e two approaches – the one described in the first sign language-specific lexicon and structure (D5.4) and a newly proposed approach based on AMR and landmarks. We conclude with brief insights on the next steps which are to be taken before finalising the SL lexicon(s) and the synthesis pipelines,

which will be presented in deliverable D5.6 (Final Sign language-specific lexicon and structure) which is due in month 36 (at the end of the project).

Revision History

Version #	Implemented by	Revision Date	Description of changes
V0.1	Dimitar Shterionov, Irene Murtagh, Bram Vanroy, Rahul Agrahari	24/04/2023	First draft
V0.2	Dimitar Shterionov, Irene Murtagh, Bram Vanroy, Rahul Agrahari	27/04/2023	Second draft - after addressing most of the comments of reviewers.
V0.3	Dimitar Shterionov, Irene Murtagh, Bram Vanroy, Rahul Agrahari	28/04/2023	Most of comments addressed
v0.4	Dimitar Shterionov, Irene Murtagh, Bram Vanroy, Rahul Agrahari	30/04/2023	After addressing comments by Lorraine

The SignON project has received funding from the European Union's Horizon 2020 Programme under Grant Agreement No. 101017255. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the SignON project or the European Commission. The European Commission is not liable for any use that may be made of the information contained therein.

Approval Procedure

Version #	Deliverable Name	Approved by	Institution	Approval Date
V0.1	D5.5	Shaun O'Boyle	DCU	25/04/2023
V0.1	D5.5	Marcello Scipioni	FINCONS	28/04/2023
V0.1	D5.5	Vincent Vandeghinste	INT	26/04/2023
V0.1	D5.5	Gorka Labaka	UPV/EHU	25/04/2023
V0.1	D5.5	John O'Flaherty	MAC	28/04/2023
V0.1	D5.5	Horacio Saggion	UPF	27/04/2023
V0.1	D5.5	Irene Murtagh	TU Dublin	28/04/2023
V0.2	D5.5	Ellen Rushe	TCD	28/04/2023
V0.1	D5.5	Jorn Rijckaert	VGTC	28/04/2023
V0.1	D5.5	Henk van den Heuvel	RU	28/04/2023
V0.1	D5.5	Lisa Rombouts	KU Leuven	27/04/2023
V0.1	D5.5	Rehana Omardeen	EUD	25/04/2023
V0.2	D5.5	Mirella De Sisto	TiU	26/04/2023
V0.2	D5.5	Lorraine Leeson	TCD	28/04/2023

Acronyms

The following table provides definitions for acronyms and terms relevant to this document.

Acronym	Definition

AMR	Abstract Meaning Representation
RRG	Role and Reference Grammar
KFS	Keypoint value Frame Sequences
GUI	Graphical User Interface
API	Application Programming Interface
MF	Manual Features
NMF	Non-Manual Features
ISL	Irish Sign Language
BSL	British Sign Language
NGT	Dutch Sign Language (Nederlandse Gebarentaal)
VGT	Flemish Sign Language (Vlaamse Gebarentaal)
LSE	Spanish Sign Language (Lengua de Signos Española)
LS	Logical Structure
PoS	Part Of Speech
IPA	International Phonetic Alphabet
Term	**Definition**
MediaPipe	A tool for pose estimation

Table of Contents

1. Introduction

Within this deliverable we outline the progress related to the creation of sign language (SL) specific lexicon and structure. It covers the development of an approach to **automate** the construction of the SL lexicon as well as the related pipelines for synthesis.

1.1. Prior challenges

We faced two main challenges in the first phase of creating SL lexicons. The first challenge was the interdependence between Role and Reference Grammar (RRG) and Sign_A (Murtagh, I.E., 2019). RRG + Sign_A would provide a robust syntactico-semantic representation, where the semantic Logical Structures (LSs), are defined in RRG and Sign_A, representing signs as "words", is embedded within these LSs.[1] Sign_A representations are then used to drive the 3D virtual signer (see Section 4 and D5.7: A Planner for Translating from Sign_A to BML-based Script). However, processing RRG and Sign_A independently (e.g. in order to be able to update either a LS without having to consider Sign_A, or vice-versa) and in parallel (e.g. in order to optimise the computational process) is not a straightforward process as it would imply that semantic and syntactic representations are constructed interdependently. For example, the semantic representation across languages remains the same while the syntax often differs, thus in the case of a translation, one needs to retain the semantic representation but change the syntax. The second challenge is the complexity related to automatically building lexemes and morphemes purely based on syntax. That is, to build the lexeme repositories and the morpheme store for each language would require a substantial amount of manual work, additional linguistic research into the sign languages involved, and a standardised methodology.

To overcome these challenges we focused our efforts on developing (sub)systems to automate the processes of (i) creating a language specific SL lexicon and (ii) X-to-sign synthesis.

1.2. Summary of contributions

- To alleviate the first challenge, we developed a method to automatically generate a different (not RRG) semantic structure, called abstract meaning representations (AMR) (Banarescu et al., 2013) which can not only be derived independently from a syntactic representation, but can be

[1] RRG and therefore the LSs are language agnostic; the Sign_A lexicon is language specific.

generated through an adaptation of the mBART model, which drives the InterL-E. (see Section 2.2)

- We built a tool to automate the process of lexicon creation for both Sign_A-based lexicon and a pose-based lexicon. This pose-based lexicon consists of sequences of keypoints or landmarks indicating the movements of the left and right hands, the fingers, the body and the face for a given sign. The lexicon entries can be derived from a video of one or multiple signs; sign labels can be input either manually or from existing corpora. The alignment of these sign labels with video frames can also be manually adapted. (Discussed in detail in Section 2.4).
- In our RRG+Sign_A approach, we have implemented a phonetic inventory using Sign_A XML specifications and a morpheme inventory. Our initial draft focuses on ISL.

1.3. Link with other deliverables

Draws from	Feeds into
D5.4 First Sign language-specific lexicon and structure	D4.4 Second distributional intermediate representation based on embeddings - InterL-E
D4.7 Second Routines for transformation of text from and to InterL	D4.2 Second symbolic intermediate representation - InterL-S
D4.12 Second adaptable pipeline for training and updating the InterL	D4.8 Final Routines for transformation of text from and to InterL
	D4.10 Final Routines for transformation of SL representations from and to the InterL
	D3.3 Linguistic description for ISL, BSL, VGT, NGT and LSE
	D4.5 A hybrid intermediate representation
	D5.2 Final version of virtual character
	D5.6 Final Sign language-specific lexicon and structure

2. Automating the lexicon construction

As defined in D5.4, our initial approach relies on using Sign_A to build the lexicon architecture for each of the supported sign languages (SLs) – ISL, BSL, NGT, VGT and LSE – where an RRG representation captures the semantics of an utterance through a language agnostic LS and Sign_A is the formalism which

documents and represents the movement, orientation, location and configuration of the hands, together with the various other non-manual features (NMF) computational phonological parameters, e.g facial movement, etc., along a timeline and at a specific location.

In particular, D5.4. describes the SL lexicon as follows: *The SL lexicon will function as a repository, which will be used for the storage of the various phonemes, morphemes and lexemes of each SL, together with their associated metadata. … The SL lexicon will act as a repository that we will input into and also that we will draw from, to provide information pertaining to each of the SLs catered for within the SignON project. The repository will also allow for the storage of new signs that are not yet catered for by the system.* [Lexicon function, D5.4., Section 2.1]

According to D5.4., the lexicon is centred around the Lexeme repository and the morpheme store: "*The lexeme repository maintains the NMFs and manual features (MF) lexemes. We use the context of an utterance to decipher whether an item should be placed within the morpheme store or within the lexeme repository of the Sign_A framework architecture. An item may exist within the morpheme store and also exist within the lexeme repository depending on its context within any given sentence. ISL morphemes, which demonstrate grammatical function, but lack any conceptual meaning will be placed within a morpheme store, while ISL lexemes or those morphemes that function in grammatical terms while also exhibiting conceptual meaning will reside within a lexeme repository.*"

The phonetic inventory contains the phonemes of the associated SL. With regard to lexicon entries, these fall in several lexicon categories. These are based on RRG following the Aktionsart (Bache, C., 1985.) classification. The SL lexeme repository would contain SL Verbs, SL Classifiers and SL Nouns. The XML structure, as can be seen in D5.4. Table 6, 7, etc. consists of the logical structure together with some other information. The logical structure includes the translations which include temporal information, location information, manual and non-manual features. The other information includes the Gloss, the IPA, the Verb number, PoS., person (verb), tense verb and the verb itself. Following is an example:

Sentence (English): 'I really love my job'

Partial XML verb lexicon entry including a logical structure:
<ISLGlossTranslate="LOVE" IPA="/lʌv/" LogicalStructure="<TNS:PRES:[LOVE <MF><NMF><LOCATION>(1sg, JOB)]>;" NumberVerb="sg" P.O.S="PlainVerb" personVerb="3rd" EnglishTranslation="love"/>

Where
<MF>, <NMF>, <LOCATION> constitute the Sign_A entry for the specific logical structure or lexeme.

Below are the MF and NMF:
Manual Features (<MF>):

```
<HAND>
   <dh>"right"</dh>
   <ndh>"left"</ndh>
</HAND>
<HS>
   <HSMode>unique</HSMode>
   <HSID>
         <value>24</value>
   </HSID>
</HS>
<AM>
   <Spatial>
         <SOURCE>"alocus"</SOURCE>
         <GOAL>"blocus"</GOAL>
         <EDti></EDti>
         <EDtn></EDtn>
         <TLti></TLti>
         <TLtn></TLtn>
   </Spatial>
</AM>
<PO>
   <p2>
         <p2_i>
               <EDti></EDti>
               <EDtn></EDtn>
         </p2_i>
         <p2_n>
               <EDti></EDti>
               <EDtn></EDtn>
         </p2_n>
         <TLti></TLti>
         <TLtn></TLtn>
   </p2>
</PO>
```

Non Manual Features (<NMF>):

```
<HEAD>
         <HEADMODE>”nod”</HEADMODE>
         <EDti></EDti><EDtn></EDtn>
         <TLti></TLti><TLtn></TLtn>
</HEAD>
<MOUTH>
         <MOUTHMODE>”smile_teeth”</MOUTHMODE>
```

```
        <EDti></EDti><EDtn></EDtn>
        <TLti></TLti><TLtn></TLtn>
</MOUTH>
<MOUTHING>
  <VERB_ONE_TO_ONE>
        <VERBIPA>"/lʌv/"</VERBIPA>
  </VERB_ONE_TO_ONE>
</MOUTHING>
```

Where some of these features assume one of a set of predefined values. E.g. <HEADMODE> and <MOUTHMODE>, where <HEADMODE> can be one of *nod, shake, tiltR, tiltL, turnR, turnL* and <MOUTHMODE> can be one of *neutral, open_wide, close_tight, smile_teeth, smile_teeth_wide, smile_closed, round_open, round_closed*. In Section 3 of D5.4 all Sign_A XML specifications are defined.

2.1. Sign_A entries extracted from pose estimates

The Sign_A formalism captures the NMF and MF, timing and location in an organized, structured way, but up till now, these had to be encoded manually. Starting from the assumption that these can be extracted from a higher-level spatio-temporal representation, an automatic, data-driven approach was implemented. We developed a lexicon extraction tool that, using MediaPipe (Lugaresi et al. 2019), extracts keypoints (spatial information) for consecutive frames (temporal information) and defines a specific Sign_A XML specification.

Sign_A is a framework that brings a linguistically motivated structure in sign representations. These representations are constrained according to SL-specific definition. As such, the task we solve is given an image or a video segment of a specific phonetic or morphological entry, convert the keypoints into a Sign_A XML specification so that these can be used in generating the body forms/shapes and movements by the avatar. A SignON team member, Lianne Quigley, who is an ISL signer and member of the Irish Deaf community, created 79 handshape images, from three different perspectives, namely from directly in front of the signer, from the side and from above. These were fed into the MediaPipe code base, which outputs a Sign_A XML specification. These XML specifications are used to populate the phonetic inventory for ISL. The handshape in the previous example has value 24, so in task (i) we take as input an image corresponding to handshape #24, identify the keypoints for each finger and create the Sign_A entry to be used in synthesis.

An image of a handshape (handshape #24):

Keypoints extracted from the image for all 21 points on hands (represented as an array):

```
[
        0:[ 6.63172126e-01,  7.10766435e-01,  2.67463378e-07],
        1:[ 6.15590811e-01,  6.85928643e-01, -3.75310816e-02],
        2:[ 5.91790318e-01,  6.18236721e-01, -5.81773669e-02],
        3:[ 6.34787381e-01,  5.58782697e-01, -7.38005564e-02],
        4:[ 6.85658455e-01,  5.36735654e-01, -8.96745473e-02],
        5:[ 5.91368496e-01,  5.08963048e-01, -5.06331027e-02],
        6:[ 5.60158014e-01,  4.25143063e-01, -7.66929910e-02],
        7:[ 5.41753173e-01,  3.64334226e-01, -9.66980606e-02],
        8:[ 5.28034568e-01,  3.09874326e-01, -1.13829672e-01],
        9:[ 6.36550128e-01,  4.85982448e-01, -5.16903475e-02],
        10:[ 6.19968772e-01,  3.81075293e-01, -7.58756548e-02],
        11:[ 6.10644460e-01,  3.09733927e-01, -9.55618396e-02],
        12:[ 6.01814032e-01,  2.43597001e-01, -1.11373171e-01],
        13:[ 6.80629253e-01,  4.96327847e-01, -5.61577417e-02],
        14:[ 6.83983386e-01,  3.98706704e-01, -8.47036839e-02],
        15:[ 6.85288370e-01,  3.29685330e-01, -1.09220862e-01],
        16:[ 6.84431553e-01,  2.64789224e-01, -1.26710400e-01],
        17:[ 7.23720193e-01,  5.28980792e-01, -6.28421232e-02],
        18:[ 7.55629420e-01,  4.58020866e-01, -9.11954269e-02],
        19:[ 7.76506543e-01,  4.07025397e-01, -1.09061413e-01],
        20:[ 7.91929722e-01,  3.57126355e-01, -1.22012593e-01]
]
```

Sign_A representation (XML):

```
<HSID>
    <value>ISL_24</value>
    <f1Shape>
        <f1Shape_i>
              <tip>
                    <x>0.5280345678329468</x>
                    <y>0.3098743259906769</y>
                    <z>-0.11382967233657837</z>
              </tip>
              <dip>
                    <x>0.5417531728744507</x>
                    <y>0.36433422565460205</y>
                    <z>-0.09669806063175201</z>
```

```
            </dip>
            <pip>
                  <x>0.5601580142974854</x>
                  <y>0.4251430630683899</y>
                  <z>-0.07669299095869064</z>
            </pip>
            <mcp>
                  <x>0.5913684964179993</x>
                  <y>0.5089630484580994</y>
                  <z>-0.05063310265541077</z>
            </mcp>
            <EDti />
            <EDtn />
      </f1Shape_i>
      <f1Shape_n>
            <EDti />
            <EDtn />
      </f1Shape_n>
      <TLti />
      <TLtn />
   </f1Shape>
   <f2Shape>
      <f2Shape_i>
            <tip>
                  <x>0.6018140316009521</x>
                  <y>0.24359700083732605</y>
                  <z>-0.111373171210289</z>
            </tip>
            <dip>
                  <x>0.6106444597244263</x>
                  <y>0.30973392724990845</y>
                  <z>-0.09556183964014053</z>
            </dip>
            <pip>
                  <x>0.6199687719345093</x>
                  <y>0.38107529282569885</y>
                  <z>-0.0758756548166275</z>
            </pip>
            <mcp>
                  <x>0.6365501284599304</x>
                  <y>0.4859824478626251</y>
                  <z>-0.051690347492694855</z>
            </mcp>
            <EDti />
            <EDtn />
      </f2Shape_i>
      <f2Shape_n>
            <EDti />
            <EDtn />
      </f2Shape_n>
      <TLti />
      <TLtn />
   </f2Shape>
   <f3Shape>
      <f3Shape_i>
            <tip>
                  <x>0.6844315528869629</x>
                  <y>0.2647892236709595</y>
                  <z>-0.12671039998531342</z>
            </tip>
            <dip>
```

```
                <x>0.6852883696556091</x>
                <y>0.3296853303909302</y>
                <z>-0.10922086238861084</z>
            </dip>
            <pip>
                <x>0.6839833855628967</x>
                <y>0.39870670437812805</y>
                <z>-0.08470368385314941</z>
            </pip>
            <mcp>
                <x>0.6806292533874512</x>
                <y>0.49632784724235535</y>
                <z>-0.05615774169564247</z>
            </mcp>
            <EDti />
            <EDtn />
        </f3Shape_i>
        <f3Shape_n>
            <EDti />
            <EDtn />
        </f3Shape_n>
        <TLti />
        <TLtn />
    </f3Shape>
    <f4Shape>
        <f4Shape_i>
            <tip>
                <x>0.7919297218322754</x>
                <y>0.3571263551712036</y>
                <z>-0.12201259285211563</z>
            </tip>
            <dip>
                <x>0.7765065431594849</x>
                <y>0.40702539682388306</y>
                <z>-0.10906141251325607</z>
            </dip>
            <pip>
                <x>0.7556294202804565</x>
                <y>0.4580208659172058</y>
                <z>-0.09119542688131332</z>
            </pip>
            <mcp>
                <x>0.7237201929092407</x>
                <y>0.5289807915687561</y>
                <z>-0.06284212321043015</z>
            </mcp>
            <EDti />
            <EDtn />
        </f4Shape_i>
        <f4Shape_n>
            <EDti />
            <EDtn />
        </f4Shape_n>
        <TLti />
        <TLtn />
    </f4Shape>
    <tShape>
        <tShape_i>
            <tip>
                <x>0.6856584548950195</x>
                <y>0.5367356538772583</y>
```

```
                        <z>-0.0896745473146438 6</z>
                </tip>
                <dip>
                        <x>0.634787380695343</x>
                        <y>0.558782696723938</y>
                        <z>-0.07380055636167526</z>
                </dip>
                <pip>
                        <x>0.5917903184890747</x>
                        <y>0.6182367205619812</y>
                        <z>-0.05817736685276031 5</z>
                </pip>
                <mcp>
                        <x>0.6155908107757568</x>
                        <y>0.6859286427497864</y>
                        <z>-0.03753108158707619</z>
                </mcp>
                <EDti />
                <EDtn />
        </tShape_i>
        <tShape_n>
                <EDti />
                <EDtn />
        </tShape_n>
        <TLti />
        <TLtn />
    </tShape>
    <tOverLap>
        <tOverLap_i>
                <EDti />
                <EDtn />
        </tOverLap_i>
        <tOverLap_n>
                <EDti />
                <EDtn />
        </tOverLap_n>
        <TLti />
        <TLtn />
    </tOverLap>
    <tPalm>
        <tPalm_i>
                <wrist>
                        <x>0.6631721258163452</x>
                        <y>0.7107664346694946</y>
                        <z>2.6746337766780925e-07</z>
                </wrist>
                <EDti />
                <EDtn />
        </tPalm_i>
        <tPalm_n>
                <EDti />
                <EDtn />
        </tPalm_n>
        <TLti />
        <TLtn />
    </tPalm>
</HSID>
```

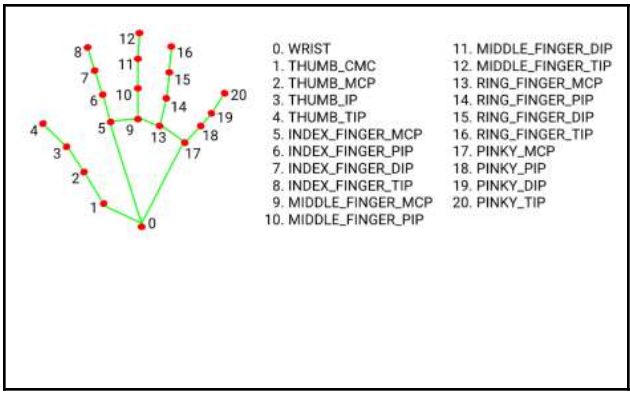
Figure 2.1. An example of the conversion steps for handshape 24 from image into Sign_A.

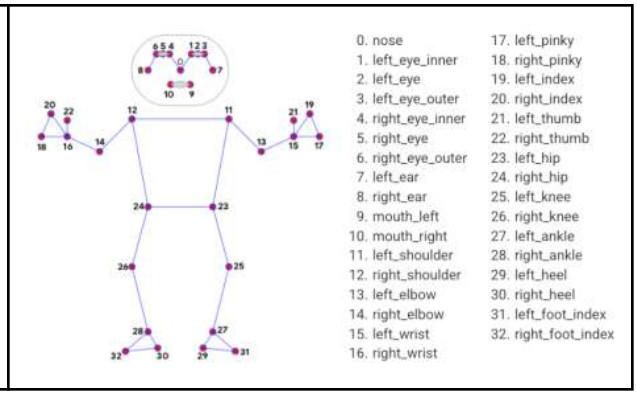
Quigley also created a number of videos pertaining to the ISL NMF parameters, e.g movement of the eyebrows, nose, cheeks, etc., which were inputted into a separate MediaPipe code base. The XML specifications that are captured here are then used to populate the ISL morpheme inventory. We also gather morpheme inventory XML specifications for ISL mouth patterns.

2.2. Lexical entries directly from pose estimation:

In Sign_A, the SL representation is broken down into MFs, NMFs, location and temporal information, such as the hand shape (represented in XML as <HS></HS>), the dominant hand (represented in XML as <dh></dh>) and so on. Aiming to simplify both the automation of lexicon generation as well as the synthesis process, we create a lexicon of pairs of a sign and a sequence of keypoint values for every video frame. For simplicity, we will refer to those as keypoint frame sequences (KFS). The KFSs are the results of MediaPipe processing a video corresponding to a particular sign. These pairs are then stored in a database. The signs are derived from already established and verified lexicons (dictionaries).[2] This approach currently handles established lexicons. However, using the tool that is described in Section 2.3, a user can record or upload a video of a new sign from a productive lexicon (together with its label).[3]

We differentiate between four tiers – left hand, right hand, body/torso and face. We created a KFS structure[4] containing the numerical values of the 3D positions of each keypoint within the four tiers: for each hand, we record the values for the 21 keypoints (see Figure 2.2 a), for the body estimate – the values for the 33 keypoints (see Figure 2.2 b), and we capture the 468 keypoints on the face mash.





[2] At the moment of writing this document the NGT and VGT dictionaries are being processed.
[3] When it comes to extending the pose-based vocabulary, the process is rather simple (see Section 2.3). However, we ought to note that it is still a matter of how the synthesis pipeline would use new lexicon entries. This is currently outside of the scope of this document and we will not discuss it further.
[4] Within the python code, this is a dictionary, which later is converted into an entry within a JSON object.

Keypoints on the hands	Pose/body keypoints

Figure 2.2. Keypoints for the hand(s) and the body

Each of these tiers is associated with an independent entry within a JSON object which is stored, together with other information, in a MongoDB database[5]. In particular, the information we store in the database is:

id	The unique identifier of the entry. Numerical value.
sign	The sign label. A string type value.
vid_path	The video path, which is needed for validation, verification and monitoring purposes. A string type value, a local path.
is_master	Whether the current record is the first one for this entry. Boolean value.
is_merged	Whether the current record is the result of merging multiple records for the same sign. Boolean value.
sim_sign_count	Number of same signs available in the repository. Numeric value.
keypoints	A list of the keypoints for the left and the right hands, the face and the pose (body/torso estimate). A set of four arrays with numerical values.

Currently we are processing NGT and VGT lexicons. Once these are done, we will proceed to ISL, LSE and BSL.

2.3. SignON lexicon generation tool

The lexicon generation tool facilitates the upload and processing of SL videos, images and text, as well as the organisation and storage of lexicon entries. It provides the functionality to upload an SL video (containing one or more signs) along with text that represents what is being signed in the video. This text can be any string of different tokens, e.g. sign labels, glosses, words, etc. Ideally, these tokens are aligned with the signs in the SL video and are used to create the sign-KFS lexicon entry. The user can manually adapt the alignment between a sign label and the video sequence presenting this sign, by altering the time frame. Using the SL segmentation algorithm of Renz et al. (2021), a SL video is segmented into

[5] https://www.mongodb.com

potential signs. To these signs, labels or tokens are attached - a user can either upload an annotation or translation text file, or type the labels. Once the video is segmented, a proposed alignment between labels and sign segments is built. The user can then edit this alignment. The tool invokes MediaPipe to process each video segment – to detect each keypoint and extract the 3D position – for the four tiers. Once the KFS is generated, the sign (label)-KFS pair is added to the database.

When a video (or a video segment) for a sign that already exists is input, the tool will try to merge the new one with the already existing record. This is done in order to create a standard, average representation of a given sign. We employed the DBA (Dynamic Time Warping Barycenter Averaging) algorithm (Petitjean et al. 2011), as provided in a public github repository[6] All other versions are retained. The values for the fields is_master, is_merged and sim_sign_count are adapted accordingly: is_master is True for the record containing the original, unmerged entry; the is_merged is True if the record contains the merged (or average) version and the sim_sing_count will keep the count of the similar signs available in the repository.

Given a sequence of sign labels, e.g. glosses (see Section 2.4), to construct a sequence with entries from the lexicon, a single SQL query suffices. However, we ought to stress that, as these entries are built from videos or video segments with different formats, quality, signers, settings, etc., SL representations that should appear in the same sequence may differ substantially, becoming a challenge for the SL synthesis.

The lexicon generation tool is equipped with a graphical user interface (GUI) as well as an API. It is deployed as a docker container and is available on our repository.

An example of a segmented video (ready for storing in the lexicon database) is given in Figure 2.3.

[6] https://github.com/fpetitjean/DBA.

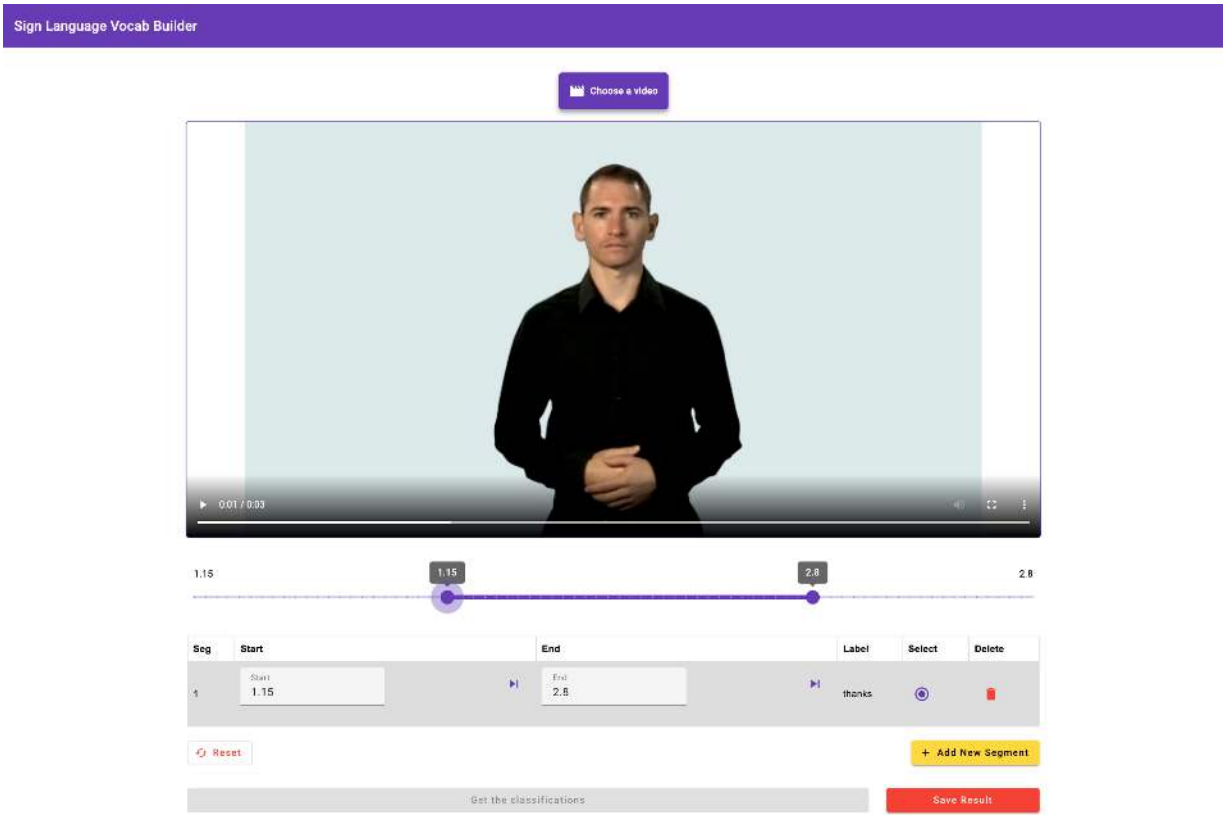


Figure 2.3. An illustration of an uploaded segmented video of the sign “thanks” in ISL. The timeframe between 1.15 seconds and 2.8 seconds is the one we want to capture, removing the frames where the signer is not performing a sign (i.e. is at-rest).

An example of a stored lexicon entry, a JSON object containing the keypoints for each of the body (pose), face, left hand and right hand, is shown in Figure 2.4.

```
_id: ObjectId('63ea40cc13c8980e520d643b')
sign: "thanks"
vid_path: "/var/folders/gx/_gh_c7796135m6wwc8hrwfr40000gn/T/tempSmallfile_1.mp4"
keypoints: Array
  0: Object
    pose: Array
    face: Array
    lh: Array
    rh: Array
```

Figure 2.4. A JSON object, representing a lexicon entry for the sign “thanks”. The details from the pose, face, lh and rh arrays are not shown as they contain the values for the three dimensions for every keypoint for every frame.

2.4. Abstract Meaning Representations

Initially, we proposed to solely rely on RRG as our framework for LS. Due to reasons described below, and as mentioned in D4.12, we are now also considering Abstract Meaning Representation (AMR; Banarescu et al., 2013) as an interlingua. In this section we first describe the difference between RRG and AMR, and motivate why AMR is a strong candidate for an interlingua.

RRG places itself in-between functionalist and structuralist theories. Like the functionalists, it considers language's main function to be communication and therefore the communicative roles of grammatical structures are central. However, it also concedes that, indeed, grammar is a system in the structuralist meaning of the word but with the caveat that this system can only be understood in light of the semantic and communicative functions. Notably for what comes next, RRG does consider syntactic form even if it only exists as directly linked with the semantic representation. AMR on the other hand aims to provide an annotation framework in which the meaning of given sentences can be represented in a labelled graph. This meaning is abstract, as the name suggests, and is completely disconnected from lexical surface form and syntax. For our research, this is important: we need an interlingua that captures the core meaning of a given sentence without being restrained by idiosyncrasies of a specific realisation. The "vocabulary" that is used in AMR (i.e., concepts and relations between them) is heavily inspired by the PropBank (Kingsbury & Palmer 2002) and is therefore also not strongly bound to a given language's surface forms as it only extracts the semantic structure. A benefit of AMR is that it was created with the goal of automatic text<>AMR generation in mind, which has led to a swift adaptation of the framework in computational linguistics and beyond. While some work on automatic RRG generation also exists, it is limited. Furthermore, the annotated datasets for AMR are larger than those that exist for RRG, which is crucial for our research; we need large datasets to train our models to achieve high performance.

In brief, we have opted to use AMR exactly because of its simplicity, its sole focus on semantics, and the feasibility of training automated systems for it because of the availability of existing datasets. We ought to stress, however, that this does not imply a replacement for RRG, given its link to Sign_A. To the contrary, we are conducting parallel research on the two formalisms. Through the comparison of these two methods to manual synthesis (we already demonstrated this with the 10 sentences, for completeness we give these sentences, together with their logical structures in Section 2.5) we will not only be able to say which one is better but also to inform the user of the complexity and time restrictions for each method relative to the achievable performance.

A demo of our implementation of a multilingual text-to-AMR model can be found here: https://huggingface.co/spaces/BramVanroy/text-to-amr. It has previously been described in D4.12. In short, we finetune an MBART model, which takes as input text and produces a linearized version of an AMR graph. For training data, we used the English-to-AMR AMR 3.0 corpus (Knight et al., 2020) and translated the input text to Spanish and Dutch with Meta AI's "No Language Left Behind" model (NLLB; NLLB Team et al., 2022). That means that for an equivalent Spanish and Dutch text, the same AMR graph should be generated. Irish is not yet included in this model. In a final iteration of the AMR model, Irish will be included by fine-tuning the model that was trained in D4.11, which includes Irish.

2.5. Examples sentences

Below we provide 10 examples of sentences for which we have the English text, their AMR representations, the VGT glosses (corresponding to the AMR) and their RRG logical structures.

- Text: Hello, nice to meet you

  AMR:

  (a / and
  :op1 (h / hello)
  :op2 (n / nice-01
  :ARG1 (m / meet-03
  :ARG1 (y / you))))

  VGT: HALLO GOED "contact" U

  LS: [[do'(1sg, say'(1sg, hello))] & [be'(3sg.neuter, nice'(do'(1sg, meet'(1sg, 2sg))))]
- Text: How are you?

  AMR:

  (h / have-manner-91
  :ARG1 (y / you)
  :ARG2 (a / amr-unknown))

  VGT: ALLES GOED Wg2

  VGT-Note: nodding

  LS: [be'(2sg, [Q-how'])]
- Text: Sorry, I do not know sign language

  AMR:

(s / sorry-01
:ARG1 (i / i)
:ARG2 (k / know-01
:polarity -
:ARG0 i
:ARG1 (l / language
:mod (s2 / sign))))

VGT: SORRY GEBARENTAAL Wg1 KENNEN

VGT-Note: NMM for negation (no gloss)

LS: [[do'(1sg, say'(1sg, sorry)] & [<neg<[know'(1sg, sign language)]>>]]]

- Text: I am trying to communicate with you via this App

  AMR:

  (t / try-01
  :ARG0 (i / i)
  :ARG1 (c / communicate-01
  :ARG0 i
  :ARG2 (y / you)
  :instrument (a / app
  :mod (t2 / this))))

  VGT: Wg1 PROBEREN WIJ-TWEE COMMUNICEREN VIA Wg APP

  LS: [do'(1sg, [be'(attempt'(1sg, via.this.app(communicate'(1sg, 2sg))))])]

- Text: How can I help you?

  AMR:

  (p / possible-01
  :ARG1 (h / help-01
  :ARG0 (i / i)
  :ARG2 (y / you)
  :manner (a / amr-unknown)))

  VGT: HOE HELPEN Wg1

  LS: [be'(do'(1sg, help'(1sg, 2sg)), [Q-how'])]

- Text: Where is the meeting?

  AMR:

(b / be-located-at-91

:ARG1 (m / meet-03)

:ARG2 (a / amr-unknown))

VGT: VERGADERING WAAR

VGT-Note: NMM for wh-question

LS: [be-loc'(<DET.definite< meeting>>, Q-where)]

- Text: When is the next meeting?

  AMR:

  (b / be-temporally-at-91

  :ARG1 (m / meet-03

  :mod (n / next))

  :ARG2 (a / amr-unknown))

  VGT: VERGADERING VOLGENDE WANNEER

  LS: [be-at'(<DET.definite< next meeting>>, Q-when)]

- Text: Excuse me, could you clarify that?

  AMR:

  (a / and

  :op1 (e / excuse-01

  :mode imperative

  :ARG0 (y / you)

  :ARG1 (i / i))

  :op2 (p / possible-01

  :ARG1 (c / clarify-10

  :ARG0 y

  :ARG1 (t / that))

  :polarity (a2 / amr-unknown)))

  VGT: SORRY Wg2 KAN MEER UITLEGGEN

  VGT-Note: NMM for polar question

  LS: [do'(1sg, say'(1sg, excuse'(2sg, 1sg)))] & [<modal.ability.can<(clarify'(2sg, that))>>]

- Text: Thank you, this has been very interesting

  AMR:

  (t / thank-01

:ARG0 (i / i)

:ARG1 (y / you)

:ARG2 (i2 / interest-01

:ARG2 (t2 / this)

:degree (v / very)))

VGT: BEDANKT INTERESSANT

LS: [do'(1sg, say'(1sg, thank'(1sg, 2sg)))])] & [do'(1sg, say'(1sg, be'(this, [very interesting']

- Text: Have a good day

  AMR:

  (h / have-03

  :mode imperative

  :ARG0 (y / you)

  :ARG1 (d / day

  :ARG1-of (g / good-02)))

  VGT: GOED DAG

  LS: [do'(1sg, say'(1sg, have'(2sg, <DET.indefinite<good day>>)))]

3. Pipelines for constructing SL representations:

In Section 2 we outlined the different methods and processes we have implemented to address the lexicon creation task. In this section we describe how the different tools and components are employed to address the overarching task of SL synthesis together with the challenges we are facing and the corresponding next steps for their resolution.

3.1. RRG+Sign_A:

As noted in Section 1, there is no clear separation between RRG and Sign_A . That is, RRG and Sign_A are intertwined and the LS should be stored in the verb repository together so that when a sentence in written text is inputted it can be decomposed and the corresponding logical structure to be constructed. The pipeline proposed is as follows:

- An English Sentence is inputted.
- The sentence is tagged with Parts of Speech (POS).
- The SL verb Logical Structures are looked up and accessed within the verb lexicon repository.

- The various arguments are then populated based on SL POS and SL morpheme and lexeme repositories.
- The RRG + Sign_A LS is produced for the associated SL.
- This is then sent to the SL realiser for SL generation.

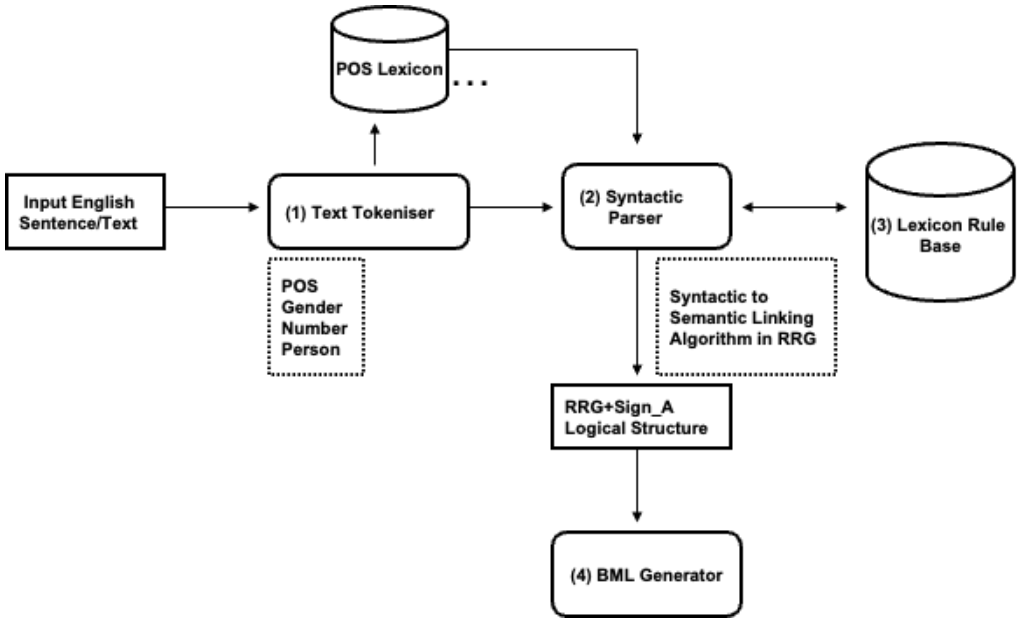


Figure 3.1. RRG + Sign_A pipeline

3.2. AMR + Keypoint frame sequences:

The alternative approach to the RRG + Sign_A, is to use AMR to construct semantic representations and keypoint frame sequences to represent the syntactic part. That is, there is a clear separation between semantic and syntactic forms which can be produced by two (sub) components: a text-to-AMR component to generate AMR given a sentence in English, Spanish or Dutch (see Section 2.3 and Deliverable D4.12) and an SQL querying system to extract lexical entries from the lexicon defined in Section 2.2. Once a sequence of lexicon entries has been established we can utilise these in the realisation of an avatar, through the approach discussed in Section 4.3 (Signing Avatars from Landmarks).There is one missing link in this pipeline, though. It is related to translating AMR predicates into SL lexicon entries. If we need to draw a parallel with RRG and Sign_A, the SL lexical elements exist within the logic structure, however, in AMR we not only have no syntactic information attached to the predicates or the AMR graph except for the AMR roles (such as ARG0, ARG1), but it also has no linear form for a specific SL. We face the following challenges: (i) how to linearize AMR into a form that

represents a grammatically correct SL sentence and (ii) how to "translate" AMR predicates into SL specific glosses that can be used to query our lexicon database. While this work lies beyond the scope of this deliverable, we briefly outline the approach we are undertaking together and the progress to date.

As was indicated in D4.12, AMR can be represented as a linearized graph, for instance as a depth-first traversal of its nodes (AMR concepts or variables) and edges (relationships). AMR concepts make use of PropBank frames, which means that they are of an English "form" (e.g. "eat-01"). We therefore need to 1. extract the concepts (not the relationships) from the linearized graph; 2. "translate" these concepts to a gloss in the target sign language (e.g. "ETEN"). The critical component of this process is the sign dictionary. Current work in this direction is limited to VGT (and should be readily extensible to NGT) because we have access to a dictionary (the VGT dictionary and the NGT signbank) that contains glosses, videos linked to these glosses, and potential translations in the corresponding spoken language. Because AMR concepts are always in English, we need to extend the dictionary with English translations. This preprocessing pipeline is outside the scope of the current deliverable, but we make use of multilingual WordNet and GPT-3.5 to find context-sensitive translations and use embedding-based disambiguation techniques. With the dictionary augmented with English translations, we can then do a reverse look-up of English word to VGT/NGT gloss. Below is an example of this pipeline.

1. Linearize AMR and extract concepts:
   eat-01 girl cookie bake-01 person have-rel-role-91 mother
2. Simplify concepts
   eat girl cookie bake mother
3. Reverse look-up in VGT-d from English translation → gloss
   ETEN-A MEISJE-B KOEK-C BAKKEN-A MOEDER-A
4. Remove regional identifier
   ETEN MEISJE KOEK BAKKEN MOEDER

Two issues remain that are currently being worked on. First, the order of these glosses is the same as the depth-first AMR traversal. This is not the correct order. We are still investigating how we can correct this order, either through statistical reordering models or by using linguistic rules (for this we could make use of the AMR relationships, e.g. "ARG0 relations always must be at the start of the segment"), and therefore do pre-reordering (as the first step). Second, we are still working on a system that can handle

words that are not found in the dictionary, i.e. English words that are not linked to a gloss. Potential solutions include on-the-fly semantic matching and for each unknown word, find the gloss that is semantically closest according to semantic similarity (e.g. fastText vector cosine similarity) or edit distance.

An alternative approach of text-to-gloss, as discussed in D4.1, is also under consideration. The two approaches will be compared during the next SignON technical meeting (4-5 May 2023, Barcelona, Spain).

The purpose of the automatic SL-specific repository is, on the one hand, to create a lexicon that can be used to synthesize a 3D virtual signer. Thus, we can start with a basic rule-based pipeline where given an AMR, a sequence of vocabulary entries is generated. These in turn are converted into their graphical representation, i.e. the Avatar.

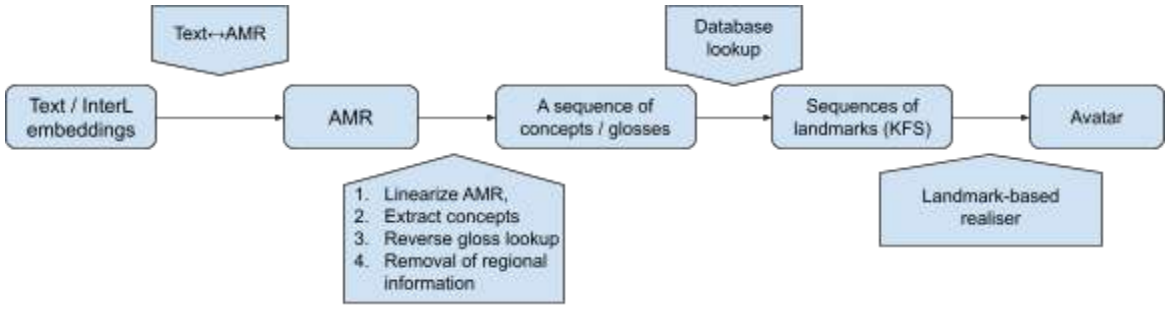


Figure 3.2. High-level text-ot-avatar pipeline illustration using AMR and KFS lexicon

4. Signing Avatar Synthesis

This section describes the most recent progress in the synthesis of the virtual signer to support the different pipelines discussed in Section 3. This progress sits within the challenge of combining *data-driven* with *procedural/symbolic strategies* to animate the signing avatar. Data-driven strategies usually lead to more natural signing, as do fully manual or semi-manual ones produced by professional animators together with expert signers. However, the latter demands a lot of resources and both do not scale well. Unlike them, procedural/symbolic strategies do scale well but usually produce less natural signing, and transitions pose an additional issue. We seek to combine both approaches taking advantage of the recent advances in Machine Learning.

In the following sections we present: 1) Progress in the procedural signing of NMFs, including mouthing, based on BML; 2) Progress in the procedural signing of both MFs and NMFs, based on an extensive dictionary NGT-SiGML, and improvements in realisation with respect to JASigning, having recently obtained access to its code; 3) Progress in the "sign animations from video signs" based on Machine Learning.

We conclude with a short section discussing the meaning of the advances.

4.1. Signing Avatar from BML

Generating the signing avatar from symbolic descriptions is the most scalable way for the production of SLs. This strategy was included in the DoA, and has been carried out during the whole project, with the preferred symbolic descriptions to be in an (extended) BML, developed within WP5T4 and appropriate to support the Sign_A based descriptions of signs, to be obtained from T3 in WP5. The choice of BML as the preferred description lies in its relationship to conversational virtual characters research, and its expression power derived from this research (including precise timings).

We discuss the most advanced ongoing work related to this type of synthesis next.

4.1.1. Non-Manual Features from BML

The animation of a virtual character performing NMFs of signs from their BML descriptions is essentially completed, to the extent that the BML descriptions are provided from previous steps of the pipeline. As an extensive Sign_A lexicon does not exist yet, the BML might need to be extended further to deal with some novel aspects that might appear when developing the extensive lexicon.

For the existing 10 phrases (see Section 2.5) for the initial tests in the five SLs SignON deals with, UPF-GTI manually generated the appropriate BML descriptions that seemed to suit those test phrases.

```
{
    type: "faceLexeme",
    start: 0.5,
    end: 4.5,
    amount: 0.8,
    lexeme: 'LOWER_BROWS'
},
{
    type: "faceLexeme",
    start: 0.3,
    attackPeak: 0.8,
    relax: 1.5,
    end: 2.0,
    amount: 0.4,
    lexeme: "LIP_CORNER_PULLER"
},
{
    type: "speech",
    start: 2.0,
    end: 2.6,
    textToLipInfo : { text: "zenk iu", phT: [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] }
},
{
    type: "speech",
    start: 3.0,
    end: 4.2,
    textToLipInfo : { text: "dhats greit", phT: [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] }
}
```

Figure 4.1. BML instructions of the NMFs for the sign *"Thank you, this is very interesting"* in BSL

The animation system is based on facial blendshapes, and builds upon previous work on puppeteering (mimicking) video input of a face in real time, including retargeting, with suitable interpolations to obtain natural dynamic facial expressions. A side-to-side comparison of the NMFs (along with the MFs) of a reference video signing *"Thank you, this is very interesting"* in BSL and the generated result can be found in the following link: https://youtu.be/TkTC9AkZCdk.

4.1.2. Mouthing from BML

Mouthing is part of NMFs, but is discussed further in this sub-subsection for convenience. Mouthing means the animation of tongue, lips and surrounding area, representing some spoken words.[7] A mouthing system based on (enhanced/extended) blendshapes has been developed. It uses the international standard phonetic representation of the spoken words, which is available in many languages. The mouthing system uses coarticulation, i.e., the mouthing depends not only on the current

[7] Mouthings are typically associated with nouns and non-inflecting verbs (Mohr, S. 2014), (Fitzgerald, A. 2014), Boyes Braem & Sutton-Spence, 2001).

phoneme but on the neighbouring ones, as it provides a more natural movement. Currently, the system, which was developed for English and Catalan/Spanish, has been tested for Dutch as well. The system is thus BML based, as well as for any language which has a phonetic dictionary for its spoken words.

While the spoken words usually have a phonetic representation, the tests carried out for mouthing in SLs have revealed that there are wide gaps in their dictionaries. For instance, Dutch mouthing was simple to provide, as a phonetic dictionary for Dutch words exists, but providing mouthing for NGT requires that the (Dutch) words being mouthed are provided as well as their timing (it is also known, e.g. Bank et al 2022, that in some cases only part of the spoken word is mouthed), in order for mouthing to be possible. Mouthing has not addressed yet stress (prosody), which is starting at the moment of writing this.

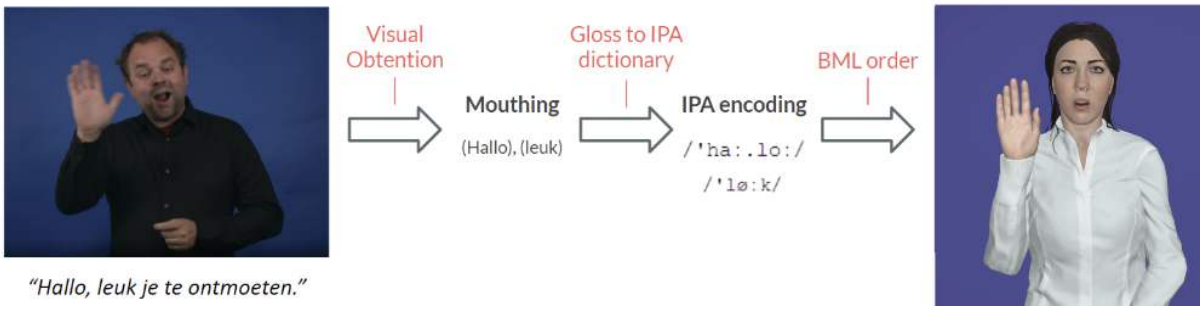


Figure 4.2. Pipeline followed to obtain the data needed to perform the mouthing of NGT sentences. The comparison between the original signer and the avatar can be found in the following link: https://youtu.be/n8reeUIaabw?list=PLdaCikBoa8nBF97lwGQrpZFzl9uyh1_iT

4.2. Signing Avatar from SiGML

As a Sign_A lexicon is not available within SignON yet, in order to realistically test the signing avatar synthesis from symbolic input, we have been using a dataset/dictionary of NGT-SiGML (for transport, Van Gemert et al. 2022; and for healthcare, Roelofsen et al 2021) to produce avatar-based SL from symbolic descriptions. While SiGML is less powerful than our extended BML, especially with regards to timing, this is a practical approach to demonstrate the on-the-fly production system developed, and compare it with existing similar alternatives. At the same time as getting the dataset from the researchers of the University of Amsterdam, we got access to JASigning code, which is commonly used to obtain output from SiGML input.

Along this line, we have carried out:

1) ***Non-Manual Features avatar production from SiGML, including mouthing improvement***. This was relatively straightforward, as the BML realiser for NMFs (including mouthing) was already available. The

mouthing improvement relates to the fact that SiGML/JASigning does not support coarticulation (see Roelofsen et al. 2021), while our realiser does. An issue with the dictionary that we have found is that quite frequently NMFs (including mouthing) are not available. For mouthing, a Dutch phonetic lexicon has been used. The following examples present a side-to-side comparison of the mouthing resulting from JASigning system (on the left), and our implementation (on the right):

- "Hallo, leuk je te ontmoeten" in NGT
  https://youtu.be/40FkJFUXNEA?list=PLdaCikBoa8nBF97lwGQrpZFzl9uyh1_iT
- "Wanneer is de vergadering?" in NGT
  https://youtu.be/2LXTKVU4cUI?list=PLdaCikBoa8nBF97lwGQrpZFzl9uyh1_iT
- "Waar is de vergadering?" in NGT
  https://youtu.be/Nth4AwkMRpI?list=PLdaCikBoa8nBF97lwGQrpZFzl9uyh1_iT

2) ***Improving handshapes not well supported in SiGML/JASigning***. Some quite critical handshapes are not well realised in SiGML/JASigning, and we have taken advantage of our work to fix this issue in a number of cases.

Figure 4.3. Comparison images of the same state of the avatar performing two different signs in NGT. “Aarde” (Earth) above, and “Bos” (Forest) below.

3) ***Adding temporal information to SiGML***. A major limitation of SiGML is not providing precise enough timing for quality signing avatar production. We have added temporal information to test the improvements in the signing quality that results. Using the example of Figure 4.4, the temporal attributes are start, attackPeak, relax, and end. Each of them indicates a specific timing of the behaviour as indicated in Figure 4.4.

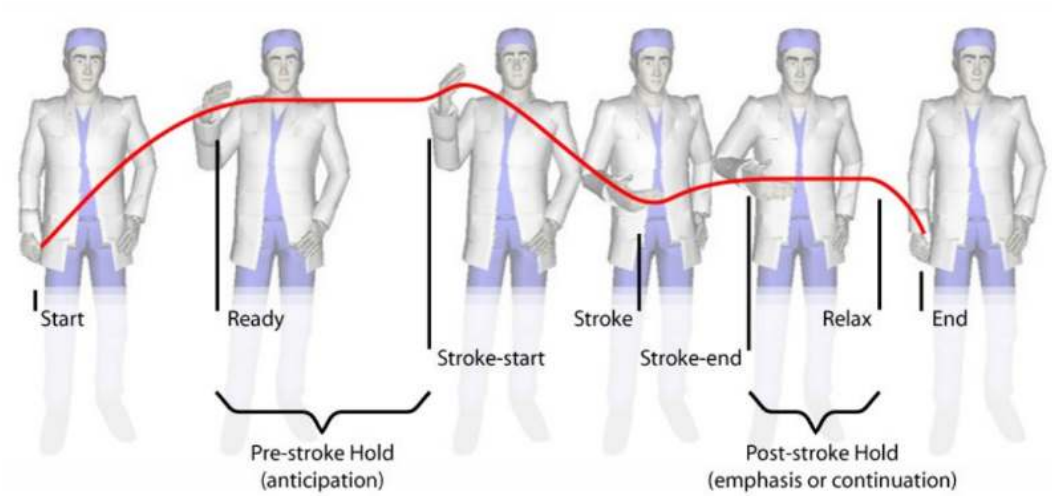


Figure 4.4. Synchronisation points of a communicative behaviour (image taken from the BML 1.0 Wiki)

4) ***Manual Features from SiGML*** *(work in progress)*. The system developed by UPF-GTI includes Inverse Kinematics (IK) support to finely adjust the MFs locations driven from BML symbolic description. Using the SiGML dictionary we have been able to expand in specific ways the amount of "instructions" (be them SiGML or BML) that can be realised/supported. Thus, the (symbolic) realiser is being extended in a pragmatic way, so that it can be used for the available dictionaries, while testing and comparison can be performed and increased the number of signs realised. As an example of the novelty just mentioned, the full animation of the sign presented in Figure 4.3 (top images, corresponding to *Aarde*) can be seen in the following link: https://youtu.be/gP641w-dVxQ

5) ***Parsing SiGML to BML*** *(work in progress)*. The realiser UPF-GTI has been building is a very flexible one; while we implement new SiGML orders, as discussed above, we map this SiGML to BML commands, so that the realiser supports these commands. In this sense, we are building a mapper from SiGML orders to BML, that would result in a functional dictionary of words to BML orders to perform animation of signs in NGT. However, a final work to build the whole parser would be required.

4.3. Signing Avatars from Landmarks

This task continues the work of animating the avatar from a video input, where a Machine Learning system for animations from videos has been created, focused on MFs. Let us discuss in more detail some aspects of the task.

4.3.1. Manual Features from Video

A major goal of the system developed, which includes editing facilities, is to be able to feed a database with high quality animations of signs (from the videos of the signs). The system already exists for some time, and its current focus is on MFs. We are still working on improving it by implementing recent state-of-the-art strategies (such as those related to Zhou et al *CVPR2021*, or recent advances in MediaPipe).

A recent related task that has been requested by some partners to support one of SignON pipelines is to use the IK subsystem to convert landmarks from a lexicon with sign-landmarks[8] entries into animations. This is a work in progress.

4.3.2. Addition of Non-Manual Features to Manual Features from Video

To make the most of the "video to signing avatar" system, which is currently focused on MFs, UPF-GTI have started to create an easy to use editor to manually manage the addition of NMFs to the MFs animations (estimated via ML) integrated in the system so that the outcome is a full sign with both MFs and NMFs. To facilitate the use of the NMFs editor, it includes quite a few usual presets of NMFs (for a question, for moods, etc.). This task is in progress.

The view of Figure 4.5 shows the new window with the new functionalities. A new timeline is needed to locate the NMFs created using the MFs keypoints as reference from the timeline above. A closer view of the timeline is presented in Figure 4.6, where we can differentiate three instructions based on Action Units (Lip Puckerer, Outer Brows Raiser, Eyes Closed) each one displayed in a different colour indicating that it influences one part of the face or another. Furthermore, a side panel is given to edit some properties of the NMF (see Figure 4.7). These orders will then be converted into the BML input that we are using to synthesise the animation of the avatar.

[8] Where the signs are in fact sign labels, e.g. glosses.

Figure 4.5. NMFs editor view, included in the same application presented in previous SignON deliverables.

Figure 4.6. Closer look at the timeline of NMFs.

4.3.3. Other related aspects

It is worth remarking that work on *Non-Manual Features from Video*, complementing the existing system focused on MFs has started. It consists of ML estimation of NMFs from video input (in a similar way as Wongpatikaseree et al. 2021). In the last weeks this has been on hold, to push more strongly on the other aspects previously mentioned towards first completion, but at time of writing, this work has resumed.

Work has not yet started on Sign Coarticulation, i.e., to produce natural transitions between precomputed animations, for instance, by removing the movement to the default pose to initiate a new sign. Coarticulation of facial expressions and mouthing is another aspect. This work is expected to commence in month 30 of the project.

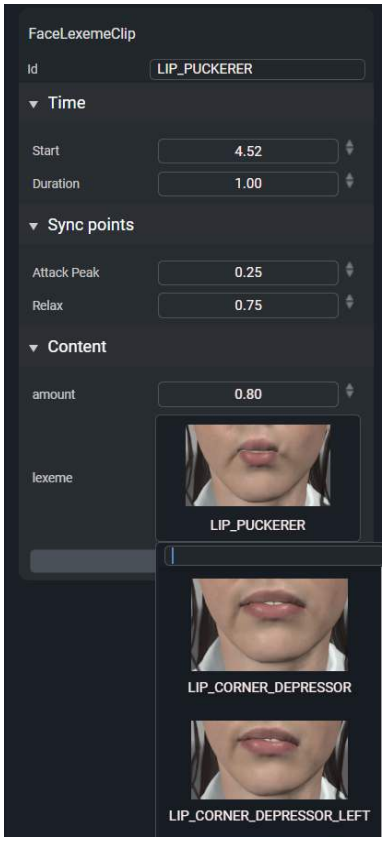


Figure 4.7. A view of the side panel to edit NMFs.

4.4. Discussion

In the recent months UPF-GTI has made significant progress, showing results that can be tested with signers, integrating MFs and NMFs, which correspond to phonological dictionaries in some SLs (specifically, NGT, which is one of the target languages of SignON). Compared to a previously existing solutions, we can show the advances that our approach and systems can provide.

This work maintains the re-usability and scalability aspects of our approach. For instance, we expect that the NGT related work can be easily extended to other SLs, when phonological dictionaries with some XML description become available. On the other hand, we have shown how to reuse, and advance, existing work based on SiGML (and improved some aspects of the code associated with SiGML, JASigning, to which we have recently had access).

A third important aspect is that this work provides alternative routes for the different synthesis pipelines that SignON is developing, especially the integration of a procedural aspect in the current “video to signing avatar” system, which provides more flexibility.

5. Conclusions

This deliverable presents the progress on the creation and utilisation of SL-specific lexicons. Following deliverable D5.4., we faced challenges related to automation. We then worked on two different approaches to tackle these challenges. The first approach is to populate Sign_A lexical and morphological repositories by exploiting landmarks from the MediaPipe software. The second approach is to build lexical entries as sign-landmark frame sequences, where each landmark frame sequence is a set of landmarks for the left hand, the right hand, the body / pose and the face, extracted by MediaPipe. To facilitate the construction of these lexicons, we developed a tool accessed by an API or by a graphical user interface. This tool supports the input of video and images, the automatic detection of signs, i.e. sign segmentation, sign classification, sign aggregation and manual timeframe correction. In this second approach, a MongoDB database holds the range of entries and can be queried at inference time to get the sequences of landmarks that correspond to a sequence of signs, i.e. a sign sentence. We are currently developing a landmark-to-avatar realizer that can generate an avatar signing the corresponding SL sentence, given a sequence of landmarks.

The specific languages we have focused on are: (i) English, Spanish and Dutch (text-to-AMR), where Irish will be supported once the InterL-E for Irish has been validated; (ii) Irish Sign Language (ISL) for the RRG + Sign_A approach and (iii) NGT and VGT for the AMR + landmark frame sequences.

Next steps include:

- Populating the Sign_A repositories and store;
- Solving the challenges in the AMR-based translation pipeline related to linearisation and reordering;
- populating the pose-based lexicon for the remaining sign languages;
- post-processing of the resulting landmark frame sequences to smooth transitions and reduce noise.

We stress that, as the entries in the pose-based lexicon are built from videos or video segments with different formats, quality, signers, settings, etc., SL representations that should appear in the same sequence may differ substantially, becoming a challenge for the SL synthesis and requiring special post-processing, as noted in the fourth point above. This is one of our major focal points and a priority among the next steps.

References

Bache, C., 1985. *Verbal aspect: A general theory and its application to present-day English*. Syddansk Universitetsforlag.

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., & Schneider, N. (2013). Abstract Meaning Representation for Sembanking. *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186. https://aclanthology.org/W13-2322

Bank, R., Crasborn, O. A., and van Hout, R. (2011). Variation in mouth actions with manual signs in Sign Language of the Netherlands (NGT). *Sign Lang. Linguist*. **14**, 248–270. doi: 10.1075/sll.14.2.02ban

Boyes Braem, Penny & Sutton-Spence, Rachel (eds.) (2001). *The hands are the head of the mouth: the mouth as articulator in sign languages.* Berlin: Signum.

Fitzgerald, A. 2014. A Cognitive Account of Mouthings and Mouth Gestures in Irish Sign Language. Unpublished PhD. School of Linguistics, Speech and Communication Sciences, Dublin: Trinity College Dublin

Foley, W. A., & Van Valin, R. D. (1984). *Functional syntax and universal grammar*. Cambridge University Press.

Kingsbury, P., & Palmer, M. (2002, May). From TreeBank to PropBank. *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*. LREC 2002, Las Palmas, Canary Islands - Spain. http://www.lrec-conf.org/proceedings/lrec2002/pdf/283.pdf

Knight, K., Badarau, B., Baranescu, L., Bonial, C., Griffitt, K., Hermjakob, U., Marcu, D., O'Gorman, T., Palmer, M., Schneider, N., & Bardocz, M. (2020). *Abstract Meaning Representation (AMR) Annotation Release 3.0* (p. 153936 KB) [Data set]. Linguistic Data Consortium. https://doi.org/10.35111/44CY-BP51

Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.L., Yong, M.G., Lee, J. and Chang, W.T., 2019. *Mediapipe: A framework for building perception pipelines.* arXiv preprint arXiv:1906.08172.

Mohr, S. 2014. *Mouth Actions in Sign Languages: An Empirical Study of Irish Sign Language*. Berlin: De Gruyter

Murtagh, I.E., 2019. *A linguistically motivated computational framework for Irish sign language* (Doctoral dissertation, Trinity College Dublin. School of Linguistic Speech & Comm Sci. CLCS).

NLLB Team, Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E.,

Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Gonzalez, G. M., Hansanti, P., … Wang, J. (2022). *No Language Left Behind: Scaling Human-Centered Machine Translation* (arXiv:2207.04672). arXiv. https://doi.org/10.48550/arXiv.2207.04672

Petitjean, F., Ketterlin, A. and Gançarski, P., 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern recognition*, *44*(3), pp.678-693.

Renz, K., Stache, N.C., Albanie, S. and Varol, G., 2021, June. Sign language segmentation with temporal convolutional networks. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2135-2139). IEEE.

Roelofsen, F., Esselink, L., Mende-Gillings, S. and Smeijers, A., 2021, July. *Sign language translation in a healthcare setting*. In Proceedings of the Translation and Interpreting Technology Online Conference (pp. 110-124).

Van Gemert, B., Cokart, R., Esselink, L., De Meulder, M., Sijm, N. and Roelofsen, F., 2022, June. First Steps Towards a Signing Avatar for Railway Travel Announcements in the Netherlands. In Proceedings of the 7th International Workshop on Sign Language Translation and Avatar Technology: The Junction of the Visual and the Textual: Challenges and Perspectives (pp. 109-116).

Wongpatikaseree, K., Hnoohom, N., Yuenyong, S. and Jaidee, S., 2021, November. Facial Action Units Recognition using Deep Learning Model with Bottleneck Features. In *2021 25th International Computer Science and Engineering Conference (ICSEC)* (pp. 318-323). IEEE.

Zhou, Y., Habermann, M., Habibie, I., Tewari, A., Theobalt, C. and Xu, F., 2021. Monocular real-time full body capture with inter-part correlations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4811-4822).

