# A software editor for the AZVD graphical Sign Language representation system

**Michael Filhol, Thomas von Ascheberg**
CNRS, LISN, Université Paris–Saclay
Orsay, France
michael.filhol@cnrs.fr, ascheberg@lisn.fr

**Abstract**

Based on real spontaneous productions by signers, AZVD is a graphical Sign Language representation system designed to maximise its potential for adoption by the signing community. Additionally, it is kept entirely synthesisable by construction, i.e. any AZVD content determines a signed output, which can be rendered through an avatar for example. This paper reports on the implementation of a software prototype developed to support AZVD editing, and the current extent of AZVD graphics integration. The point is to allow users to experience and discuss the AZVD approach, and ultimately assess it as a standardised graphical form for Sign Language representation.

**Keywords:** Sign Language, graphical form, writing system, AZVD

## 1. Introduction

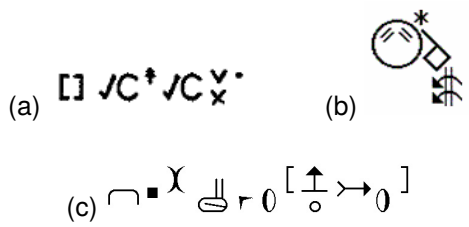
Languages that have a written form are often equipped with software assisting in various types of processing, the first of which being text editing. In contrast, sign languages (SLs) have no written form. While video is often used as a default substitute, it cannot be considered equivalent: its storage is heavy, and it is comparatively laborious to edit, index or query. Moreover, interpreting any of its contents is subject to real time, whereas reading allows to scan and capture multiple parts of the input freely. It also prevents anonymity, which is a significant limitation when considering information and opinion circulation on the internet for example.

First we show a few systems proposed and techniques used by SL users to work around this problem. Then we present the recent "AZee Verbalising Diagram" (AZVD) approach to graphical SL representation, designed to be synthesisable by signing avatars and maximise adoptability by the users. We follow by describing a software editing prototype that we developed to test the system and ultimately evaluate it.

## 2. Verbalising diagrams

To work around or address the lack of adopted SL writing system, some scripts were developed, three of them shown in fig. 1. Some were created for scripting purposes, for linguistic annotation or computer synthesis. Some have claimed a writing system status or potential. But none is adopted by the wide communities of language users (Grushkin, 2017; Kato, 2008).

And yet, there are clues that the need for some form of writing exists. Deaf people and translators




(a) Stokoe's notation (Stokoe et al., 1965)
(b) Sign Writing (Sutton, 2014)
(c) HamNoSys (Prillwitz et al., 1989; Hanke, 2004)

Figure 1: Examples of graphical systems designed for sign languages

also take notes or prepare SL discourses by drawing diagrams that somehow capture their structure, meaning or content in some more or less readable form (Athané, 2015). These diagrams exhibit various arrangements of icons, text, drawings, lines and arrows. An example of such "verbalising diagram" (VD), from the corpus built by Filhol (2020a), is given in fig. 2. It represents an LSF production of 56 s, signed after the diagram was drawn, with the following meaning:

> Atoms are very small particles, composed of a nucleus and electrons (elementary negative electric charges). Atoms are electrically neutral because their nucleus holds as many positive charges as electrons do negative charges. Groups of atoms are called molecules. Ions are atoms or molecules with electrons gained or lost from the action of neighbouring atoms. Ions are therefore electrically charged.



*Proceedings of the 11th Workshop on the Representation and Processing of Sign Languages*, pages 77–85
25 May 2024. 

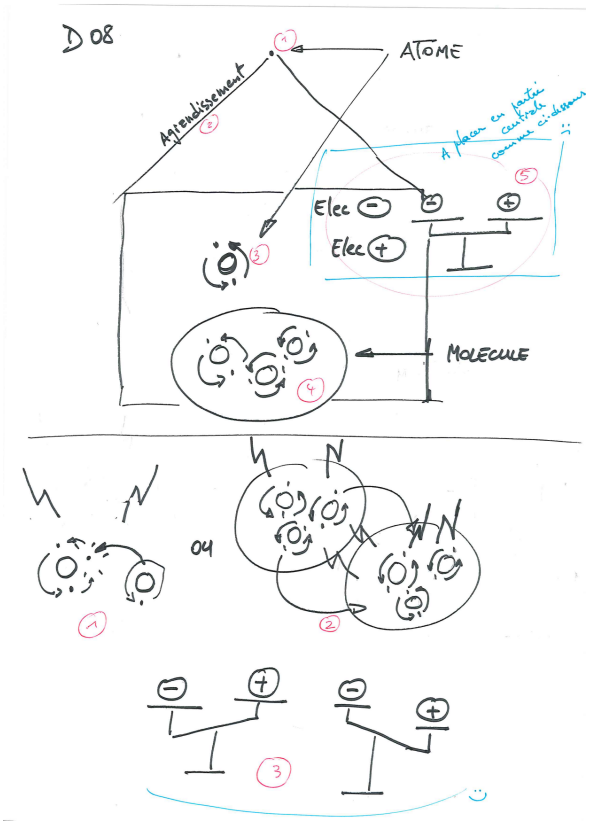
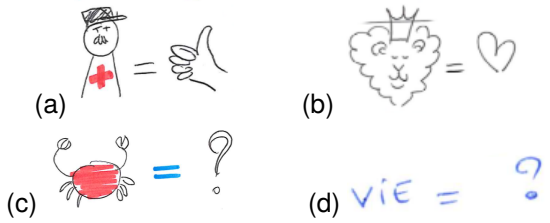



Figure 2: Example of verbalising diagram

VDs are spontaneous productions in the sense that the contained graphics follow no predefined set of rules. This usually makes parts of them readable only by the original author. In other words VDs do not strictly determine the signed form to produce to read them out, so they cannot be viewed as synthesisable input to, say, signing avatars. This is in contrast with a shared property of standardised writing systems, which we consider powerful as content becomes exchangeable in an anonymous, light-weight and editable fashion.

However, after collecting a corpus of VDs from French Sign Language (LSF) users, regularities have been reported both across diagrams and across authors (Filhol, 2020a), to the point where some VD layouts or icons with an identifiable meaning have a systematic signed equivalent when read out by their author. An example is given in fig. 3, where the same '=' symbol is consistently used between a left- and a right-hand side—say $L$ and $R$—to mean that $R$ is a state or property of $L$. This is almost systematically signed with $L$ and $R$ in this order with a form of assertion. Another, more trivial example is also visible in the same figure: the '?' symbols here consistently stand for the sign commonly glossed "QUOI" (French for "what").

The spontaneity of the VD representations and the presence of regularities already in the productions led us to propose that a standardised graph-




(a) Fidel Castro's health = good (F. C. is well)
(b) lion = nice (the lion is nice)
(c) cancer = '?' (what is cancer?)
(d) life = '?' (what is life?)

Figure 3: VD exemplars using the "equal" sign (with meaning in context)

ical script inspired by them could be experienced as a more natural way of representing signed content, hence increase its *adoptability* (Filhol, 2020b). Such a script would include the regular VD layouts when observed, while completing the set for language coverage in a way that it remains *synthesisable*.

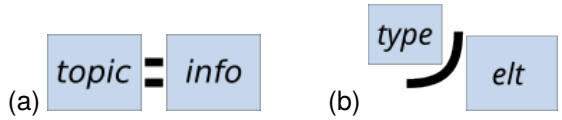
The recent AZVD proposition is a first attempt at satisfying these two features. The next section presents it in further depth.

## 3. AZVD

AZVD is a formal graphical system combining 2D symbols, borrowing from the observed spontaneous ones like that in fig. 3. Similarly to the mathematical script in which atomic tokens (e.g. numbers, variable names) and operators (e.g. unary '!', binary '+', ternary '$\Sigma$') recursively combine to grow formulae of arbitrary size, AZVD allows to build recursive diagrams to represent SL utterances of arbitrary size. To make diagrams synthesisable, every symbol or layout defined in the graphical system is *mapped* to a signed output in a given language, making use of the nested arguments as appropriate. The specification of this output is done with AZee expressions or templates.

AZee is a formal SL representation system used for synthesis with avatars. It defines the notion of *production rule*, i.e. a strong association between a meaning and an articulated form in a SL. The set of production rules for a language is called its *production set*. AZee has already proven efficient in terms of language coverage (Challant and Filhol, 2022) and feasibility and quality of synthesis (Mcdonald and Filhol, 2021) in LSF.

Some of the regular VD patterns directly correspond to LSF production rules. For example the semantic relationship between elements $L$ and $R$ carried by the "equal sign" layout (fig. 3) is exactly the meaning carried by AZee expression `info-about(topic=`$L$`, info=`$R$`)`. An AZVD map-




(a) `info-about` (*topic*, *info*)
(b) `instance-of` (*type*, *elt*)

Figure 4: AZVD layouts with variable sections, and their AZee expression mappings

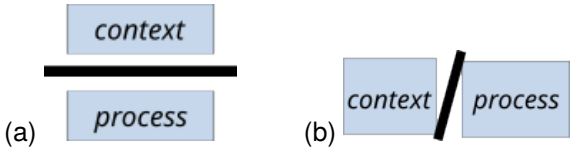



Figure 5: Two AZVD layout variants mapping to `in-context` (*context*, *process*)

ping is therefore warranted between the layout in fig. 4a, with variable parts $L$ and $R$, and that AZee expression with the same variable parts. Others can involve more elaborate AZee constructions.

More layouts are then added for AZee coverage when no sufficient spontaneous regularity was observed in VD. This applies to the set of rules supporting the basic sign vocabulary (every sign needs an icon), but also the combining, structuring rules. For example, no stable VD layout was established for `instance-of`[1], so we created a layout for it, shown in fig. 4b.

AZVD also allows to map a similar AZee output from multiple graphical layouts, as was observed in VDs. For example, straight separation bars corresponding to the meaning and form of AZee rule `in-context`[2] are commonplace in VDs (one horizontal instance is visible in fig. 2). They can be oriented in different ways, hence the definition of two *variants* of the same mapping (fig. 5).

Recursively then, any full AZVD combination determines a single AZee expression output. And since any AZee expression determines a single SL production as a result, AZVD guarantees that every diagram ultimately determines a single read-out, making it synthesisable in a testable manner.

For example, fig. 6 shows the AZVD for the full 2B-JP entry of the *40 brèves* corpus[3] (Filhol and Challant, 2022), whose signed production lasts 27 seconds and meaning is the following:

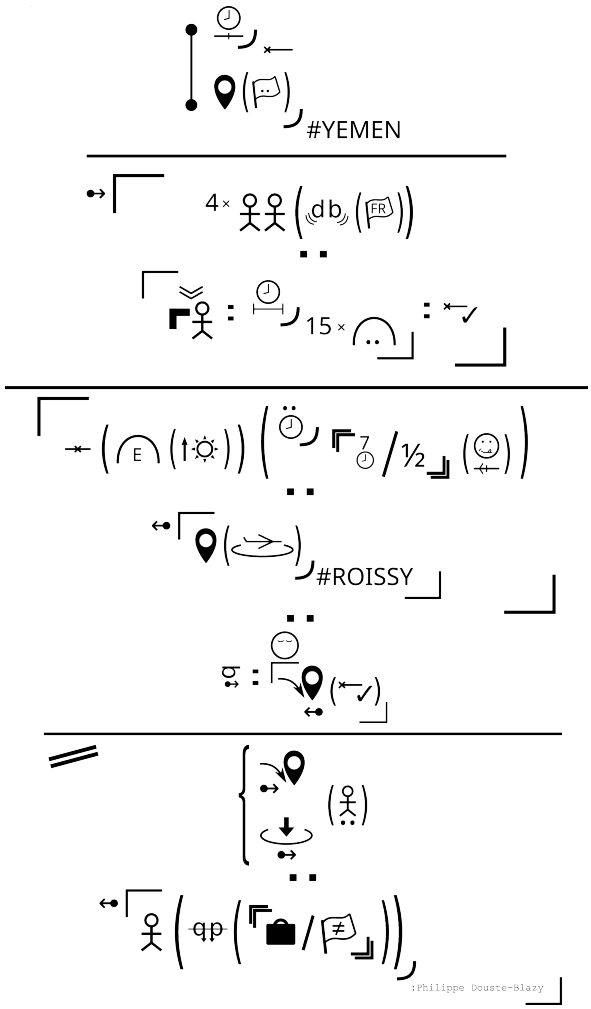



Figure 6: Example of AZVD

> The four French tourists who were kidnapped 15 days in Yemen arrived on Wednesday, shortly before 7:30am, at Roissy airport, met by Minister of Foreign Affairs Philippe Douste-Blazy.

The AZVD in the figure exactly maps to the reference AZee expression, which in turn evaluates to a timeline specifying the necessary articulations for an avatar to render the same utterance.

AZVD is therefore a graphical system that is both synthesisable in principle and built with a method intended to maximise its adoptability. Testing synthesisability is verifying that an avatar animation can be rendered automatically from the expressions generated by composed graphical input, i.e. essentially AZee synthesis. To test adoptability, we ultimately need to place the system in the hands of users and involve the community in an iterative evaluation and improvement loop. To allow this process, the first step was to develop a software editor, able to assist in drawing AZVD in a controlled manner.

[1] Meaning of `instance-of` (*type*, *elt*) : *elt*, understood as an instance of *type*.

[2] Meaning of `in-context` (*context*, *process*): event or state *process*, which happened in situation *context* or after *context* has happened.

[3] Each of the 120 entries consists in a video LSF translation for a French news item, and the AZee expression that represents it. https://www.ortolang.fr/market/corpora/40-breves

Figure 7: Screenshot of the AZVD editor: icon and layout menu on the left; main editor canvas in the middle; generated AZee output on the right

## 4. A software editor for AZVD

To enable testing of the AZVD proposition, we developed a software editor supporting the creation and manipulation of AZVD content. As we expect it to evolve with the AZVD system itself, we made the two following design choices:

- develop the editor as a web application to avoid requiring any installation or updating process on the user end, and enable instant deployment across all users on server upgrades;
- keep AZVD-side specifications separate from the server and load them dynamically on browser page load, in order to allow as much AZVD evolution as possible without changing the core application code.

After an overview of the chosen user interface, this section explains what the necessary AZVD components are, and how they are specified separately.

### 4.1. User interface

Inspired by the common WYSIWYG[4] interfaces to similar graphical content creation tasks, such as *Qt Designer* (windowed GUI design) or *Dia* (2D diagram drawing), we opted for a window layout with a central *canvas* to edit the AZVD content, and elements available in a left-hand *menu* to populate it with through drag-and-drop operations. We also added a right-hand *output panel* to display the generated AZee expressions, as we have stated the goal and benefit that every diagram determines one, and one only. This output synchronously reacts to every change on the canvas. A screenshot of the interface is given in fig. 7.

The top-level unit of AZVD specification is the left-hand menu object, which must contain information on both what to draw when inserted on the canvas and what AZee expression to generate as output. This is close to what has been called an "AZVD mapping" up to here. The way to specify them for the editor is described in the next section.

[4] "what you see is what you get"

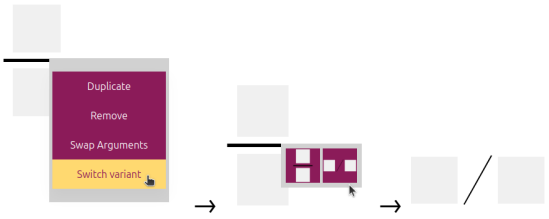



Figure 8: Switching from vertical to horizontal variant, specified in the same JSON spec file

### 4.2. Menu entries

As introduced earlier, adding, removing or changing AZVD mappings should be possible outside of the server implementation, whether to specify a graphical layout, icon or AZee output. At the moment this is done by providing JSON specification files, dynamically populating the menu on page load according to the specified content.

Each JSON file lists at least a description of a graphical layout (or fixed icon) and an AZee template to output when used on the canvas. To relate variants in the interface and pack them in a single menu entry, we allowed several layouts to be specified together in the same spec file. For example, both variants in fig. 5 can be specified together, in this case both mapping to the same parameterised AZee expression. This allows easy switching between variants of elements already on the canvas, as illustrated in fig. 8.

This explains how menu entries are created. The next two sections respectively deal with how to specify graphical layouts and the corresponding AZee output expressions.

### 4.3. Graphical layouts

In the general case, layouts are composed of one or more elements, each of which can be fixed graphics (e.g. an icon or line) or a variable part. Specification of a graphical layout is a problem of alignment and scaling of those contained elements. For example, the layout of fig. 5a is a group of three elements (two nested diagrams $context$ and $process$, and a horizontal bar between them), aligned vertically through the centre, equally spaced, and the width of the middle bar constrained to be a little longer then the widest of the other two by a few points.

To do this, we first defined primitive element types to include in a layout:

- scalable graphics, rendered as specified directly inline with standard SVG code;
- text, which is rendered as a label verbatim in

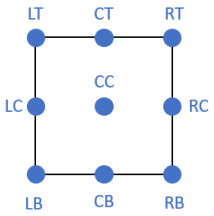



Figure 9: Generic element hotspots, named after horizontal and vertical positions relative to bounding box (L=left; T=top; C=centre; R=right; B=bottom)

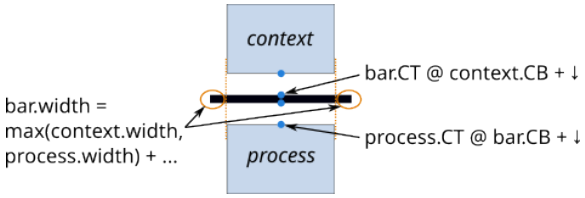



Figure 10: Specification of the layout in fig. 5a

the diagram, creating its own graphical bounding box;

- *drop zones*, which stand for the named variable parts of the layout, e.g. $context$ and $process$ above, to be filled for a complete diagram—they are rendered as plain grey boxes when empty.

Secondly, we implemented a relative positioning system based on generic hotspots assumed for any layout element, shown in fig. 9. Each new element in a layout is inserted by positioning one of its hotspots relatively to another's, with or without an offset expressed in absolute terms or relatively to other elements' sizes. Scaling or resizing elements is also possible, on either or both axes, proportionally or separately, in absolute terms or relatively to other elements' sizes.

Fig. 10 collects all the specifications required for the example layout of fig. 5a. It includes two "CT under CB" positioning constraints, which stack and centre the elements one under the previous, and one width scaling constraint on the horizontal bar, expressed as a function of the other elements' widths. This way, the width of the bar will adjust dynamically when the content of either drop zone is modified.

### 4.4. AZee output

Every layout must specify the AZee expression it maps to, so that the AZee output panel be immediately updated with the new content when the layout is placed on the canvas. This can be a simple case of a fixed expression from a fixed layout, or one with variable parts like those in figures 4 and 5.

If the layout contains variable parts supported by drop zones, the output depends on their content, provided by further graphics filled in by the user. The AZee output then depends on the expressions that this content generates. In such case, the output specification can refer to the AZee for the nested content using a provided operator and the names of the zones, just like figures 4 and 5 used the same variable names as the labels in the corresponding layouts. Empty drop zones (incomplete diagrams) will generate a placeholder label to stand for the missing content, and dropping any graphical content in an empty drop zone will automatically update the output by expanding the placeholder to reflect the change.

## 5. Evaluation of current progress

The editor has reached a technically usable state, and we are gradually providing AZVD mappings for the AZee production set of LSF, as explained in section 3, to populate the menu. Straight away however, we note that graphical coverage of the entire production set is an unreasonable target to condition first tests on.

One reason is the size and open-endedness of the sign vocabulary. Looking at the *40 brèves* corpus alone, which serves as the AZee reference for LSF today (totals 1 hour of AZee-encoded LSF discourse), we find that out of the 858 distinct production rules applied, 768 are defined with no mandatory arguments[5]. Besides, we believe that users should be given priority to propose the icon graphics, debate choices[6] and possibly feed back to one another after some practice. Therefore, the effort to create enough individual icons to cover any significant portion of the vocabulary appears greater than we can afford without a dedicated team. It would also only serve as a kick-start proposition to be entirely reviewed anyway. But to provide enough vocabulary for the sake of demonstration, we decided to choose 5 entries of the corpus of which to cover the vocabulary entirely, namely 1A-OC, 1B-JP, 1O-VF, 1R-JP and 2B-JP. This represents a vocabulary set of 114 signs.

To insert signs—or indeed any signed piece of discourse—with no graphical solution yet, we created an alternative to AZVD mappings, namely

[5]This is to us the best characterisation of a vocabulary—or *lexical*—unit in AZee: a signed production that can be delivered without contextual input (a "citation", "canonical" form).

[6]For example, should it capture the meaning (promote a logographic symbol) or the articulated form (compose a phonographic encoding) of the represented sign? We have already documented the fact that spontaneous productions exhibit a logographic prevalence overall, but not an exclusive one (Filhol, 2020b).

*AZee boxes*. An AZee box can be dropped on the canvas instead of a regular graphical layout, and filled with AZee code, which will directly serve as its own mapped output. For a sign without an icon defined, an AZee box can therefore be used, filled with a simple named rule application, looking essentially like a gloss until an icon is defined. An example is visible at the bottom of fig. 6 (“`:Philippe Douste-Blazy`”), which is a name-sign for a prior member of the French government, for which we thought creating an icon was unnecessary.

Vocabulary signs aside, we are left with the production rules requiring at least an argument when applied. In our *40 brèves* count, that remainder consists in 90 rules of the featured set:

- 12 types of pointing gestures (e.g. using index or hand sweep);
- 20 rules representing objects referred to as “classifiers” in the literature (e.g. `prf-flat-surface`, `prf-person-standing`);
- 58 recursive rules of various arities (unary rules like `with-worry`, binary ones like `info-about`, etc.).

The 58 recursive rules are the most interesting to cover as they are those building up the backbone structure of the discourse expressions. They typically have higher frequencies, and constitute a set that is much less open-ended than the sign vocabulary, in other words less subject to subsequent extensions. This is in a sense a more *grammatical* set, and securing mappings for it is a lot more stable an achievement than covering any vocabulary subset. Incidentally, and contrary to the lexical set known to be more significantly different between SLs, recent experiments seem to indicate that this set may be mostly transparent across different SLs (McDonald et al., 2024 (to be published). It is something of interest if we later want to consider AZVD beyond its application to LSF.

We have covered all rules of that set with a working graphical layout and AZee mapping in the editor, except:

- 5 rules related to classifier use and geometric placement in signing space (`landmark-in-place`, `place-object`, `mult-around`, `mult-in-a-row`, `deploy-shape`);
- 4 rules supporting the logic for numbers above 20 (built with multipliers and sums) and doubled letters in fingerspelling—although these rules will not require graphics because numbers and words to fingerspell will appear spelt out in diagrams without being broken down (but an extension to the AZee output generation language from these text units will be necessary).


Figure 11: AZVD mapping for “`pointage index(`$target$`)`”


Figure 12: Basic AZVD point layouts

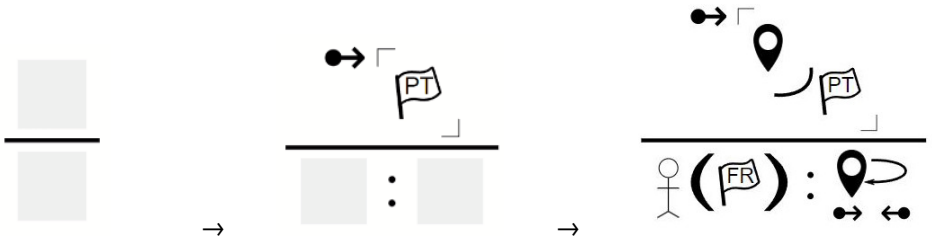
Let us now consider the pointing rules. Their signatures all resemble vocabulary signs, only they usually require a $target$ argument of type POINT, and sometimes other geometric arguments, e.g. for orientation in a plane. Accounting for such a rule with AZVD is therefore comparable to finding an icon for a dictionary sign, only a non-optional variable part must be part of the layout. Fig. 11 shows the layout defined for `pointage index`, by far the most frequent: 256 occurrences in the corpus, over 4 times as many as the second-ranked, and indeed the 6th most frequent rule all together. Notice that it features a variable part, awaiting a point expression.

To enable filling such point arguments, we defined three more mappings, from the symbols shown in fig. 12 to the most basic and frequent point expressions in the corpus. These are `^Mssp` (neutral, central point of the signing space at about a forearm’s length of the signer’s abdomen), and `^Lssp` and `^Rssp` (points on either side of it, left and right respectively).

The more complex geometric point constructions or signing space references will require an AZee box at this point of our progress, and providing it the AZee code explicitly. The remaining 20 production rules in the above count, related to classifiers, have also not been accounted for yet. We come back to those in the prospects below.

In summary, aside from complex number and geometric constructions, we have reached most of the grammatical production set for LSF already. For instance, the 2B-JP entry of the *40 brèves* corpus, shown in fig. 6, is fully editable within the program.

Fig. 13 illustrates a few steps of an AZVD construction, with the corresponding AZee output for each, for an LSF production meaning “French person returns from Portugal”, with Portugal located on the right-hand side of the signing space first, and the person returning to the left-hand side at he end. A recorded video of the whole process is available at `https://zenodo.org/records/10890951`. Note how every drop, move or swap action on the canvas updates the AZee output accordingly.




```
:in-context
  'context
  %placeholder%
  'process
  %placeholder%
```

```
:in-context
  'context
  :about-ref
    'pt
    ^Rssp
    'info
    :Portugal
  'process
  :info-about
    'topic
    %placeholder%
    'info
    %placeholder%
```

```
:in-context
  'context
  :about-ref
    'pt
    ^Rssp
    'info
    :instance-of
      'type
      :lieu
      'elt
      :Portugal
  'process
  :info-about
    'topic
    :side-info
      'focus
      :une personne
      'info
      :France
      'info
      :retour
        'src
        ^Rssp
        'dest
        ^Lssp
```

Figure 13: Progressive construction of an AZVD with the editor. NB in French: “lieu” = place, location; “une personne” = a person; “retour” = return.

## 6. Conclusion and future work

After reviewing a spontaneous practice of drawings to represent SL, we presented the AZVD system aiming to propose a graphical system both synthesisable and adoptable by SL users. We followed by presenting a software editor developed to support creation and editing of diagrams in the AZVD format.

The point of the editor in the long run is to allow users to apprehend the AZVD approach, evaluate its adoptability, and involve them in the system’s evolution as much as they would like to. But measuring adoptability with users through the editor can only be conducted reliably if it is fit to support AZVD manipulation transparently enough in the first place. An incomplete or non-ergonomic, counter-intuitive interface can indeed lead to rejection of AZVD as a whole even if the cause is the editor alone.

So to avoid this bias, we must separate the evaluation of the editor and that of AZVD as a scripting system. We will do so by taking a first step testing the application essentially as an AZee editor first. That is, measure how AZee experts feel assisted in the task of writing and reviewing AZee expressions. Any piece of AZee not covered with AZVD graphics can still be expressed in AZee code inside AZee boxes, which AZee coders would have done anyway without the editor. Reaching a positive evaluation on that aspect would constitute evidence that the interface and features of the editor provide enough comfort and assistance to allow users to direct their judgement at the manipulated script, not the manipulation tool.

Now even with a good editor, limitations to AZVD remain which should also be addressed. The most limiting factors are the missing layouts for the geometric rules and constructions (involving classifiers)

and the sign vocabulary (lexical set). Each of these two aspects represents a work prospect to increase the scope of AZVD graphics.

Geometric/classifier constructions were postponed mostly because a parallel work to encode the *Mocap1* corpus (LIMSI, 2020) with AZee expressions is in progress. This substantial work should result in a more stable reference for AZee representation of those constructions, which was to us an interesting contribution to wait for before defining a graphical layer for them. However, it is already clear that their infinite range in signed locations, paths, dynamics and classifier options does not come from an ever-growing set of ad hoc rules, but from the generative power and combinatorics of a limited set. We therefore believe tentative solutions should be in reach, similar to those addressing the grammatical set, only certainly requiring more layouts for native geometric objects (points, vectors, paths). This is to at least remove the constant need for AZee boxes in the diagrams, and propose a first graphical scheme to the discussion along with the other grammatical rules.

In contrast though, as explained above, it is impossible to do the same with the open-ended set of vocabulary signs. The prospect for us here, aside from keeping the possibility of glossing (a strategy well captured by AZee boxes already), is to allow to fallback on custom graphical choices, and ideally integrate a proposal and voting system, or existing lexically-oriented phonographic systems such as SignWriting or HamNoSys (fig. 1). Choices could be up- or downvoted by the community, and we would get to observe discrepancy or consensus in propositions. How variable are the logographic choices? How often do phonographic ones make spontaneous use of the existing systems? Much is yet to be learnt, on top of what VDs already exhibit, about how SL users envision scripting their language symbolically.

In the mean time, one already pictures the kind of diagrams AZVD allows to build, and notices two major differences with the prior systems. First, logography is allowed and frequently used in the graphics. We have already said that it played a major part in spontaneous VDs, while being totally absent in the other systems. Second, the diagrams exhibit the meaningful links between their constituents, reflecting the underlying structure of the utterances. This is very similar to the spontaneous VDs, which rarely present entirely separate parts, and rather keep them connected in a planar (2D) drawing. It also greatly contrasts with the other systems, which impose to follow the production sequence one lexical unit after the other, without connecting them in any meaningful way. If we trust the idea that following spontaneous practice is likely to favour adoptability, both of those properties are therefore welcome.

Finally, we would like to leverage the fact that AZVD was designed not only to maximise adoptability, but also to be synthesisable. Integrating an avatar to the interface, for example under or instead of the AZee output panel, to render the AZVD canvas content would bridge over the full pipeline from AZVD editing to dynamic display of the scripted signed discourse. We are preparing for this exciting prospect, which in our view will even allow users with no knowledge of AZee to learn AZVD directly.

This way we hope to put the system in the hands of the Sign Language community with as few obstacles as possible to appreciating the AZVD system. More than evaluating a fixed state from a single field test, we will hopefully engage users on a continuous improvement process, and fuel the discussion about graphical Sign Language writing, which is an unresolved issue yet.

## 7. Acknowledgement

This work has been funded by the EASIER (Intelligent Automatic Sign Language Translation) European project. Funding's agreement Horizon 2020 no. 101016982.

## 8. Bibliographical References

Anaïs Athané. 2015. La schématisation : un travail original de préparation à la traduction de textes vers la langue des signes française. *Double Sens*, 4.

Camille Challant and Michael Filhol. 2022. A first corpus of azee discourse expressions. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Marseille, France.

Michael Filhol. 2020a. Elicitation and corpus of spontaneous sign language discourse representation diagrams. In *Proceedings of the 9th workshop on the representation and processing of sign languages*.

Michael Filhol. 2020b. A human-editable sign language representation inspired by spontaneous productions... and a writing system? *Sign Language Studies*, 21(1).

Donald A. Grushkin. 2017. Writing signed languages: What for? what form? *American Annals of the Deaf*, 161(5):509–527. Gallaudet University Press.

Thomas Hanke. 2004. Hamnosys—representing sign language data in language resources and

language processing contexts. In *Proceedings of the workshop on the Representation and Processing of Sign Languages*, pages 1–6. European Language Resources Association (ELRA).

Mihoko Kato. 2008. A study of notation and sign writing systems for the deaf. *Intercultural Communication Studies*, 17(4):97–114.

John Mcdonald and Michael Filhol. 2021. Natural Synthesis of Productive Forms from Structured Descriptions of Sign Language. *Machine Translation*.

John McDonald, Rosalee Wolfe, Eleni Efthimiou, and Evita Fotinea. 2024 (to be published). Multilingual synthesis of depictions through structured descriptions of sign: An initial case study. In *Proceedings of the workshop on the Representation and Processing of Sign Languages*, Torino, Italy.

Elena Antinoro Pizzuto and Paola Pietrandrea. 2001. The notation of signed texts: Open questions and indications for further research. *Sign Language & Linguistics*, 4(1–2):29–45.

S. Prillwitz, R. Leven, H. Zienert, T. Hanke, and J. Henning. 1989. Hamnosys version 2.0, hamburg notation system for sign languages, an introductory guide. *International studies on Sign Language communication of the Deaf*, 5. Signum press, Hamburg.

William C. Stokoe, Dorothy C. Casterline, and Carl G. Croneberg. 1965. *A Dictionary of American Sign Language on Linguistic Principles*. Washington, DC.

Valerie Sutton. 2014. *Lessons in SignWriting*, 4th edition. The SignWriting Press.

## 9. Language Resource References

Filhol, Michael and Challant, Camille. 2022. *40 brèves*. 2, ISLRN 988-557-796-786-3.

LIMSI, CIAMS. 2020. *MOCAP1*. ISLRN 502-958-837-267-9. ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr.