Implementation of an Automatic Sign Language Lexical Annotation Framework based on Propositional Dynamic Logic

Arturo Curiel[†] **, Christophe Collet**
Université Paul Sabatier - Toulouse III
118 route de Narbonne, IRIT,
31062, Toulouse, France
E-mail: curiel@irit.fr, collet@irit.fr

Abstract

In this paper, we present the implementation of an automatic sign language (SL) sign annotation framework based on a formal logic, the Propositional Dynamic Logic (PDL). Our system relies heavily on the use of a specific variant of PDL, the Propositional Dynamic Logic for Sign Language ($\mathrm{PDL}_{\mathsf{SL}}$), which lets us describe SL signs as formulae and corpora videos as labeled transition systems (LTSs). Here, we intend to show how a generic annotation system can be constructed upon these underlying theoretical principles, regardless of the tracking technologies available or the input format of corpora. With this in mind, we generated a development framework that adapts the system to specific use cases. Furthermore, we present some results obtained by our application when adapted to one distinct case, 2D corpora analysis with pre-processed tracking information. We also present some insights on how such a technology can be used to analyze 3D real-time data, captured with a depth device.

Keywords: sign language framework, automatic annotation, propositional dynamic logic

1. Introduction

Research in sign language (SL), both from the point of view of linguistics and computer science, relies heavily on video-corpora analysis (Dreuw et al., 2008). As such, several methods have been developed over time for the automatic processing of both video or other sensor-based corpora (Ong and Ranganath, 2005). Even though these kind of research efforts are usually geared toward recognition, few work has been done in relation to the unification of raw tracked data with high level descriptions (Cooper et al., 2011; Bossard et al., 2004). This calls to a reflection on how we represent SL *computationally*, from the most basic level.

SL lexical representation research is focused on sign synthesis before than recognition. Works like (Filhol, 2009; Losson and Vannobel, 1998) present the use of geometric lexical descriptions to achieve animation of signing 3D avatars. While their approach is well suited for synthesis, it is not completely adapted for sign identification. Recognition tasks in both natural language processing and computer vision are well known to be error-prone. Also, they are highly susceptible of bumping into incomplete information scenarios which may require some kind of inference, in order to effectively resolve ambiguities. In addition, SL linguistic research has consistently shown the existence of common patterns across different SLs (Aronoff et al., 2005; Meir et al., 2006; Wittmann, 1991) that may be lost with the use of purely geometrical characterizations, as the ones needed in synthesis. This limits the application of these kind of sign representations for automatic recognition, especially since we would want to exploit known linguistic patterns by adding them as properties of our descriptions. Works like (Kervajan et al., 2006; Dalle, 2006) have acknowledged the necessity of introducing linguistic information to enrich interaction, in an effort to help automatic systems bear with ambiguity. Moreover, the use of additional linguistic data could simplify connections between lexical information and higher syntactic-semantic levels, hence pushing us closer to automatic discourse analysis. However, this has long been out of the scope of synthesis-oriented description languages.

On the side, research in SL recognition has to deal with other important drawbacks not present in synthesis, namely the use of very specialized tools or very specific corpora. This alone can severely impact the portability of a formal, computer-ready, representation out of the original research context, as it complicates the use of the same techniques across different information sources and toughens integration with new tools.

The framework described here is based on previous work presented by (Curiel and Collet, 2013) on the Propositional Dynamic Logic for Sign Language ($\mathrm{PDL}_{\mathsf{SL}}$). $\mathrm{PDL}_{\mathsf{SL}}$ is a formal logic created with the main purpose of representing SL signs in a computer-friendly way, regardless of the specific tools or corpora used in research. Such a representation can potentially reduce the overhead of manually describing SL signs to a computer, by establishing well-known sets of rules that can be interpreted by both humans and automatic systems. This could, incidentally, reduce dependency on thoroughly geometrical descriptions. Moreover, the flexibility of $\mathrm{PDL}_{\mathsf{SL}}$ lets us combine any kind of information in our descriptions; for example, we can integrate non-manual markers if we have sight and eyebrow tracking, or we can add 3D movements if we are using a depth camera.

In general, we propose an automatic SL lexical annotation framework based in $\mathrm{PDL}_{\mathsf{SL}}$ descriptions. Ideally, the system will:

- simplify the application of logical inference to recognize $\mathrm{PDL}_{\mathsf{SL}}$-described signs;

[†] Supported by CONACYT (Mexico) scholarship program.

- characterize and analyze corpora in terms of PDL_{SL} models;
- represent SL with different degrees of granularity, so as to adapt the formulae to the specific technical capabilities available in each use case.

Our framework aims to ease the integration of PDL_{SL} with various corpora and tracking technologies, so as to improve communication between different SL research teams. We expect that this will, in turn, enable the construction of both research and user-level applications in later stages.
The rest of the paper is divided as follows. In section 2., we introduce the basic notions of our formal language, applied to 2D SL video-corpora analysis. Section 3. shows how we can describe SL lexical structures as verifiable PDL_{SL} formulae. Section 4. gives a detailed description of the system's architecture. Finally, sections 5. and 6. present some preliminary results and conclusions, respectively.

2. Sign Language Formalization with Logic

The Propositional Dynamic Logic (PDL) is a multi-modal logic first defined by (Fischer and Ladner, 1979) to characterize computer languages. Originally, it provided a formal framework for program descriptions, allowing them to be interpreted as modal operators. PDL_{SL} is an specific instance of PDL, based on the ideas of sign decomposition by (Liddell and Johnson, 1989) and (Filhol, 2008). In general, PDL_{SL}'s modal operators are movements executed by *articulators*, while static postures are interpreted as propositional states reachable by chains of movements.
A propositional state will be none other than a set of distinct atomic propositions. These can be used to represent articulators' positions with respect to one another; specific configurations; or even their spatial placement within a set of *places of articulation*. Table 1 shows a brief summary of the atomic propositions defined to analyze 2D corpus data.

Symbol	Meaning
$\beta_1{}^{\delta}_{\beta_2}$	articulator β_1 is placed in relative direction δ with respect to articulator β_2.
$\mathcal{F}_c^{\beta_1}$	articulator β_1 holds configuration c.
$\Xi_\lambda^{\beta_1}$	articulator β_1 is located in articulation place λ.
$\mathcal{T}_{\beta_2}^{\beta_1}$	articulator β_1 and β_2 touch.

Table 1: Atomic propositions for PDL_{SL}

Basic movements can be described by atomic actions codifying either their direction, speed or even if they follow a particular trajectory. This is exemplified by the definitions on Table 2, which presents some of the operators used to characterize 2D corpus movements.
Both atomic propositions and actions presented in this case were chosen specifically to capture information that we are able to detect with our tracking tools. Different sets of atoms can be defined depending of the technical capabilities available to assert their truth values (*e.g.* sight direction, eyebrow configuration, hand movement, etc).

Symbol	Meaning
δ_{β_1}	articulator β_1 moves in relative direction δ.
$\leftrightsquigarrow_{\beta_1}$	articulator β_1 *trills*, moves rapidly without direction.
skip	denotes the execution of any action

Table 2: Atomic actions for PDL_{SL}

Atoms form the core of the PDL_{SL} language, which is presented below in Backus–Naur Form (BNF) by way of definitions 1 and 2.

Definition 1 (Action Language for SL Body Articulators $\mathcal{A}_{\mathsf{SL}}$)**.**

$$\alpha ::= \pi \mid \alpha \cap \alpha \mid \alpha \cup \alpha \mid \alpha;\alpha \mid \alpha^*$$

where π is an atomic action.

Definition 2 (Language PDL_{SL})**.**

$$\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid [\alpha]\varphi$$

where p denotes an atomic proposition and $\alpha \in \mathcal{A}_{\mathsf{SL}}$.

A more formal presentation of the model basis can be found in (Curiel and Collet, 2013).

3. Extending PDL_{SL} formulae to Describe Sign Language Lexical Properties

The presented PDL_{SL} language lets us easily codify individual signs by way of our logic formulae. However, during implementation, we noticed the need to extend the original formalism in order to develop a better suited characterization of more general properties. We wanted to represent lexical structures common across multiple signs. With this in mind, we extended PDL_{SL} to include *lambda expressions*, explained in (Barendsen, 1994), for variable binding. The introduced syntax is presented in definition 3.

Definition 3 (Extended PDL_{SL})**.**

$$var ::= \langle \texttt{uniqueID} \rangle \mid var, var$$

$$\varphi_f ::= \varphi \mid var \mid \neg\varphi_f \mid \varphi_f \wedge \varphi_f \mid \lambda\, var.(\varphi_f) \mid var = \varphi_f$$

where $\varphi \in \text{PDL}_{\mathsf{SL}}$.

The rules of quantification and substitution remain the same as in classic lambda calculus.
Lambdas let us describe properties over sets of PDL_{SL} atoms instead of one. For example, Figure 1 shows two french sign language (FSL) signs, $\texttt{SCREEN}_{\mathsf{FSL}}$ and $\texttt{DRIVE}_{\mathsf{FSL}}$. Both can be described as instances of the same underlying common structure, characterized by both hands holding the same morphological configuration while being positioned opposite from one another.
Their common base can be described by way of a lambda expression as shown in example 1.

Example 1 (**opposition** lambda expression)**.**

$$hands\_config = \lambda c.(\mathcal{F}_c^{\texttt{right}} \wedge \mathcal{F}_c^{\texttt{left}})$$
$$\textbf{opposition} = \lambda c.(\texttt{right}^{\leftarrow}_{\texttt{left}} \wedge hands\_config(c))$$

SCREEN$_{\text{FSL}}$ DRIVE$_{\text{FSL}}$

COMMON STRUCTURE

Figure 1: Comparison of signs SCREEN$_{\text{FSL}}$ and DRIVE$_{\text{FSL}}$ sharing the same underlying structure

In example 1, $\mathcal{F}_c^{\texttt{right}}$ means that `right` holds configuration c. Atom $\mathcal{F}_c^{\texttt{left}}$ has the same meaning, but for the `left` hand. Atom $\texttt{right}^{\leftarrow}_{\texttt{left}}$ means that `right` hand lies in direction $\leftarrow$ with respect to `left`, from the annotator's point of view. In this case we called our expression **opposition**, because both hands are in opposite horizontal positions from one another.

Once we've defined the base structure, the SCREEN$_{\text{FSL}}$ and DRIVE$_{\text{FSL}}$ signs can be easily described in the database by passing the missing arguments to our lambda expression (as shown by example 2).

Example 2 (**opposition**-derived signs)**.**

$$\text{SCREEN}_{\text{FSL}} = \textbf{opposition}(\texttt{L\_FORM})$$
$$\text{DRIVE}_{\text{FSL}} = \textbf{opposition}(\texttt{FIST\_FORM})$$

In example 2, `L_FORM` is a morphological configuration of the hand where the thumb and the index fingers are held orthogonally. Similarly, `FIST_Form` is a configuration where hand is held as a closed fist. Here we just expressed that **opposition** will substitute each apparition of its first argument with either form, so as to define two distinct signs.

We could also have described both signs as standalone, independent formulae. However, by describing the common structures across different signs, we are able to cope better with incomplete information in recognition. For example, a generic **opposition** structure with free variables will correctly hit in states where we can recognize hand positions but no hand configurations (as it's often the case). This immediately derives into a list of possible signs that could be later reduced with either further processing or with user interaction. In this scenario, standalone formulae for SCREEN$_{\text{FSL}}$ and DRIVE$_{\text{FSL}}$ wouldn't be found, since only using position information isn't enough to tell them apart.

4. Detailed Framework Architecture

The objective of the system is to take an untreated SL video input, either in real time or not, and return a set of satisfied PDL$_{\text{SL}}$ formulae. Moreover, the system has to return a PDL$_{\text{SL}}$ model representing any relevant information contained in the video as a labeled transition system (LTS). This can only be fulfilled by adapting the modeling process on-the-fly to the specific characteristics of our data. To achieve this end, our framework generalizes the original architecture proposed by (Curiel and Collet, 2013), shown in Figure 2, so as to enable module swapping depending on the technical needs presented by the inputs.

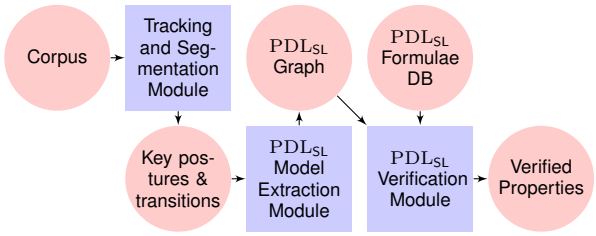

Figure 2: Block diagram of a generic PDL$_{\text{SL}}$-based SL lexical structure recognition system

In the original version, a *Tracking and Segmentation* module uses the raw data of an automatic hand-tracker on 2D corpora, like the one presented by (Gonzalez and Collet, 2011), and returns a list of time-intervals classified either as *holds* or *movements*. The aforementioned interval list is passed to the *Model Extraction Module*, which translates each *hold* and *movement* into a time-ordered LTS. In the LTS, *holds* correspond to unique propositional states and *movements* map to transitions between states. An example of the resulting LTS is shown in Figure 3.

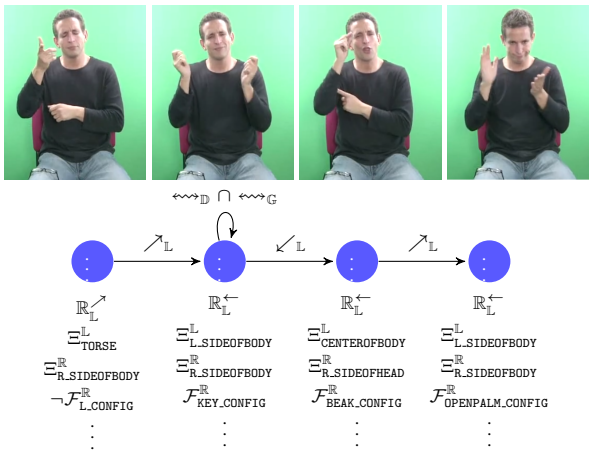

Figure 3: Example of modeling over four automatically identified frames as possible key postures

Finally, the *Verification Module* takes both the generated LTS and a database of PDL$_{\text{SL}}$ formulae to determine which of them are satisfied in the model. As each formula corresponds to a formal description of a sign or property, the module can use logical satisfaction to verify if the property is present or not in the video. The complete process is shown in Figure 4. Finally, the system maps each state

where a formula is satisfied to its corresponding frame interval, so as to generate an annotation proposition.

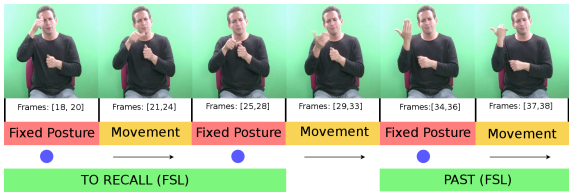

Figure 4: Example of the different layers processed by an automatic annotation system

4.1 Observer Architecture Design

In order to be able to adapt dynamically to the particular needs of the input data, we devised the observer architecture shown in Figure 5.
The main idea behind this design rests upon two axes:

- the possibility of using several tracking tools, adapted to different kinds of corpora;
- the generation of PDL_{SL} models consistent with the information generated by the different trackers.

Moreover, not only do models have to be consistent with every tracker but, as previously stated, not all trackers will give the same information nor track the same features. As such, the framework has to coordinate the loading of the proper modules depending on the corpus and the trackers. This process is entirely done by way of event-triggering. The same mechanism enables communication between modules by implementing multiple-reader/single-writer (MRSW) buffers, which allow every module to read their information but let only one of them write modifications. Each time a new modification is written in a MRSW register, an event is issued system-wide to notify of the existence of new information. This event is then available to every module listening to that register's notifications. For the sake of compatibility, modules are obliged to implement an internal listening thread which can be subscribed to the communication channels of any other module.
In general, the framework establishes development guidelines for the modules of the basic architecture, the one shown on Figure 2, so we can adapt them to specific cases without breaking compatibility. This is achieved by way of generic templates that implement the most basic functionalities of every module. These templates can later be extended to cover the specific cases arising in research; a developer can simply override the critical functionality in each template with their own code. Additionally, modules can register new events within the framework, so as to convey further information (if needed) for particular cases. As such, the system is capable of distributing self-contained, interchangeable, modules that can adapt to different situations.
The execution process is also fairly straightforward. At the beginning a *Start* event is fired-up, prompting to load both a video stream and a tracker. This corresponds to the *Tracking and Segmentation Module* on the basic architecture (Figure 2). The system chooses between the compatible video inputs and pairs the selection with the proper tracker. This is done by reading the events sent out by the loading functions. Likewise, the model construction rules are loaded after a compatible set of video/tracking inputs has been selected. In this way, we can assure that the modeling algorithm will only take in account pertinent rules, those relying on the specific features we are tracking. This mechanism avoids generating models based on hand positions, for example, if our tracker is only capable of detecting non-manuals. Once a compatible set of modules is activated, the process can continue as proposed by (Curiel and Collet, 2013).

5. Experimental Results

We obtained some preliminary results on the proposed framework by implementing the system's core and a set of minimal templates for each of the modules on Figure 2. The core contains the necessary data structures to represent both PDL_{SL} models and formulae, alongside the semantic rules necessary to acknowledge logical satisfaction.
For the creation of the module templates, we considered two possible scenarios:

- the system is being used to annotate previously captured video corpora;
- a camera as going to be used as input for real-time sign recognition.

Furthermore, we had to consider two distinct cases when treating video; whether we had 2D or 3D information available for determining relationships between hands and body. For simplicity, we worked only with hand-tracking data. Nevertheless, the addition of non-manual trackers is also a possibility, since introducing new modeling rules for non-manuals follow the same principles of the 2D to 3D transition.
Once all the framework tools were in place, we created a specific implementation for the 2D case, when tracking features over existing corpora.

5.1 Automatic Annotation in 2D Corpora

To obtain some initial results over real-world data, we developed the first modules based on the atoms originally presented with the PDL_{SL} language. Additionally, we created a property database made of PDL_{SL} formulae, adapted to be used with our tracking device. The database position in the architecture is shown in Figure 5, as the node *Lexical Formulae*. The formulae were exclusively constructed for the 2D case; this means that, for any other kind of tracking information, we would need to define new PDL_{SL} database with different properties. For tracking, we used the tracker developed by (Gonzalez and Collet, 2011), which is capable of finding 2D positions of the hands and head over SL video corpora. As for the SL resources, we used an instance of the *DictaSign* corpus (DictaSign, 2012) as video input for our system.
Since the used tracking tool is not adapted for real-time processing, the implemented tracking module just recuperates

the previously calculated information from an output file. This is done sequentially, after each successful querying of a new video frame, to simulate real-time.

To calculate the posture segmentation we used the method proposed by (Gonzalez and Collet, 2012), which is based on measuring speed-changes.

Our $\mathrm{PDL}_{\mathrm{SL}}$ description database contains four structures:

opposition. $\lambda c.(\mathtt{right}^{\leftarrow}_{\mathtt{left}} \wedge hands\_config(c))$. Hands are opposite to each other, with the same configuration.

tap. $\lambda s, w.(\neg\mathcal{T}^s_w \rightarrow [moves(s) \cup moves(w)]\mathcal{T}^s_w \rightarrow [\mathbf{skip};\mathbf{skip}]\neg\mathcal{T}^s_w)$. Hand touches briefly the other hand, only for a single state.

buoy. $\lambda s, posture.(posture \wedge [moves(s)^*]posture)$. The state of one hand remains the same over several states, regardless of the movements of the other hand.

head anchor. $\lambda s, w, posture.(\mathbf{buoy}(s, posture) \wedge \mathcal{T}^{\mathtt{head}}_w)$. One of the hands remains within the head region while the other signs.

The $posture$ variable denotes the propositional state of an articulator. The $moves(s)$ function can be interpreted as any action executed by articulator s. We omit the complete, formal definition of this operator for the sake of readability.

To measure the *hit* ratio of the system, we manually annotated the apparition of the described properties in one video within the corpora. Table 3 shows the quantity of observed apparitions of each property over the chosen video.

φ	**oppos.**	**buoy**	**tap**	**h. anch.**
Total	76	40	33	74

Table 3: Manually annotated apparitions of property formulae on one video

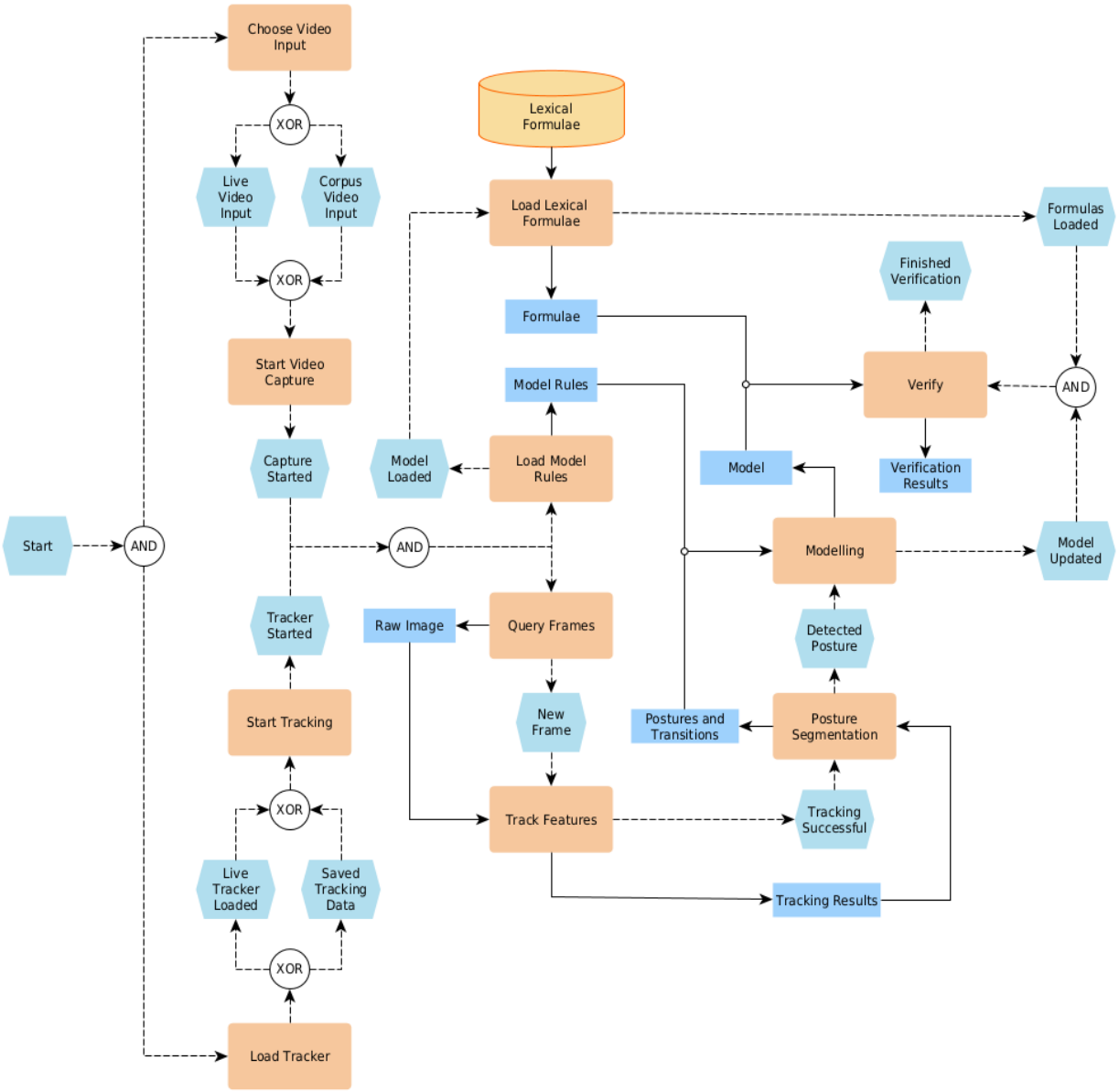

Figure 5: Information and control flow in the SL annotation framework

For each signer, the system creates a model based only on the atoms specified by the modeling rules. It then uses the created model to verify every formula on-the-fly. The execution of our algorithm over the same video rendered the results shown in Table 4.

φ	**oppos.**	**buoy**	**tap**	**h. anch.**
Total	164	248	79	138

Table 4: Total reported hits of property formulae on one video

On Table 4 we can see the total number of times each of the formulae were verified on the video, as returned by the system. We compare the human annotations with these results on Figure 6.

TOTAL OBSERVATIONS

φ	**oppos.**	**buoy**	**tap**	**h. anch.**
By hand	76	43	33	74
Automatic	164	245	79	138

Human \ Automatic	φ	**oppos.**	**buoy**	**tap**	**h. anch.**
	oppos.	67	64	10	33
	buoy	22	40	7	17
	tap	15	24	25	11
	h. anch.	23	50	13	44
	False P.	37	67	24	33

Figure 6: Formulae verification results

Figure 6 shows data from both Tables 3 and 4, alongside a *matching table* where, for each property formula, we count the quantity of times it was verified on previously human-annotated frames. Each row represents the total number of human observed apparitions of one property, while each column represents the quantity of positive verifications reported by the system for each formula. For example, cell (**opposition**, **opposition**) shows the total number of times the **opposition** formula was correctly verified on human-annotated **opposition** frames. The next cell, (**opposition**, **buoy**), holds the number of times the **buoy** property was verified on human-annotated **opposition** frames. Positive verifications can overlap, *i.e.* the system could have verified two or more formulae over the same states of the model. Therefore, a single annotation could belong to different classes. The cells on the last row of the table correspond to *false positives*, reported detections that don't overlap with any human observation.

Further analysis on the matching table is represented on Table 5, which shows the total number of correctly and incorrectly classified formulae, as well as the total mismatches.

The results show a high recognition rate for **opposition**, **buoy** and **tap**, but also a high quantity of misclassification hits and false positives. Most of the erroneous hits are due to the definitions of the properties themselves. Take, for example, **opposition** and **buoy** properties. In the video, some of the states satisfying a **buoy** could easily be classified as **opposition**. When this happens, the only thing that differentiates them, if we only have tracking information, is their movement: if a hand is moving is a **buoy**, otherwise is an **opposition**. Even though this is not always the case, sometimes the situation arises and the system confuses these properties for one another; if some of the movements of the hands are too fast, or not ample enough, when performing a **buoy**, the system interprets them as a static posture, therefore classifying some of the internal states of the **buoy** as **opposition**. This, however, doesn't impede finding the buoy, since the definition of **buoy** specifies, from the beginning, an arbitrary number of internal states, hence not affected by having found one instead of two distinct states. The opposite case might also arise, when a short involuntary movement, is interpreted by the system as an intended action instead of noise, hereafter classifying an **opposition** as a **buoy**, or even as two sequential **oppositions**. Similar arguments can be made for **tap** and **head anchor**, where movement thresholds alone can affect the form and the quantity of states on the LTS. In the future, we expect that adding new information will reduce the quantity of misclassified data, specially because this will result in a more fine-grained model from the beginning.

φ	HUMAN OBS. HIT	HUMAN OBS. MISS	ERRONEUS MATCH
opposition	67	9	107
buoy	40	3	46
tap	25	8	50
h. anchor	44	30	86

Table 5: Per-formula summary of the total number of observations found, missed and erroneously classified observations

At this stage, the system returns a list of proposed properties as result of the verification phase. What the numbers on Table 5 mean is that, in most cases, the proposed annotation will almost never return single properties but rather sets of properties. This may not be a problem with simple formulae like the ones described, but would be problematic with complete sign descriptions; there is such thing as too much information. In that case, we would need a human being to complete the classification process. This points out the need or a higher level module in charge of cleaning the annotation proposal by way of machine learning techniques.

Finally, most of the false positives that don't correspond to any overlap with human observations were caused by signer's movements without communication intent. For example, some **opposition** properties were found when a signer crossed his arms, when his hands were posed over his knees or when he assumed other natural repose positions. Similarly, some co-articulatory movements created chains of states that satisfied the formulae for **buoy** or **tap**. These cases could also be reduced with help of a higher level module or a human expert.

5.2 Extending to 3D

Currently, we are extending the system to model features tracked in 3D. We have already extended the framework to process data returned by the Kinect (Microsoft, 2013),

a motion sensing device capable of tracking 3D positions on several body articulations. Figure 7 shows the points that can be tracked by using the Kinect with it's official development kit.

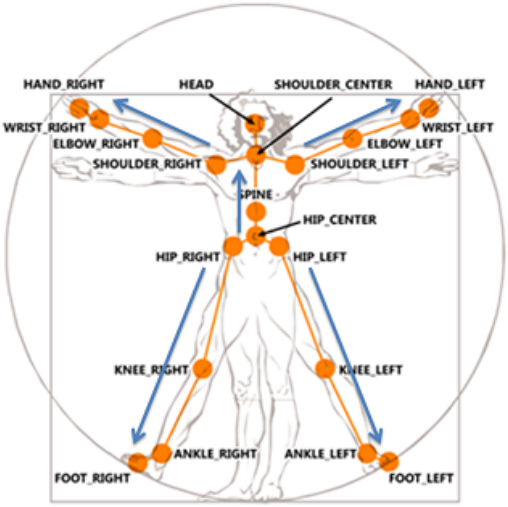

Figure 7: Points tracked by the Kinect device (Microsoft, 2013)

For the moment, we have been able to reuse the same modeling rules that we implemented for the 2D case; mainly, we have used the Kinect tracker to obtain 3D position data of hands and head, and we have projected this information in 2D. This lets us create the same kind of models we build from corpora. However, the variety of the tracked articulations and the 3D capabilities of the sensor, call for the definition of more complex atoms and lambda properties, as well as 3D descriptions of individual signs. As of 2014, work is still ongoing on the matter and has not been properly evaluated. Nevertheless, we considered important to point out that we can already exchange trackers if needed, so as to showcase the flexibility of our framework.

6. Conclusions

Here we have presented an automatic annotation framework for SL based on a formal logic. The system lets us represent SL video inputs as time-ordered LTSs by way of $\mathrm{PDL}_{\mathsf{SL}}$, a multi-modal logic. We have shown that it is possible to use the resulting graph to verify the existence of common lexical structures, described as logical formulae. Furthermore, the framework gives us the necessary tools to adapt the model generation for different corpora and tracking technologies.

From the point of view of recognition, we noticed that the quality of the tracking tools is of utmost importance for both formula definition and model generation. The low presence of information and high levels of noise immediately took a toll on verification; in some cases, we lacked enough information to distinguish between intended movements and noise. In turn, this resulted on high rejection rates of what would otherwise be considered *hit* frames.

Similarly, we noticed that modeling can be affected by the presence of low information, which can render states indistinguishable. For instance, without hand configurations every state satisfying **opposition** is, effectively, the same state. Therefore, every formula sharing the same **opposition** base would be satisfied on that single state. This could gravely affect the system's performance; in the worst case, all states could satisfy all formulae. On the other hand, a too fine-grained model can lead to a LTS that replicates the same problems we have in synthesis-oriented descriptions. In that case, we would need very specific formulae (with near to perfect SL corpora) to achieve any identification at all. Similarly, formula creation can't be neither too broad nor too specific, if we want to minimize the quantity of imperfect matches. Anyhow, one of the advantages we have by using a logical language is that we can control the granularity of information simply by defining or discarding atoms, which opens the door to the use of algorithmic techniques to control information quantity.

From the point of view of the implementation, the results of the 2D experiments show that further effort has to be put on integrating new sources of information to the system, especially if we want avoid false positives. Even though the system is in place and works as expected, the high quantity of erroneous hits reflects the gravity of the problems we can have with indistinguishable states. Further comparisons have to be done once the system completely incorporates 3D modeling, so as to measure the effective impact of additional information on verification.

Future work in recognition will be centered on implementing machine learning techniques to improve verification. Using data analysis to find relationships between detected structures, could lead us to better results even in suboptimal environments. Additionally, we would like to integrate communication with user level software like the one presented by (Dubot and Collet, 2012), a manual annotation tool. This could lead to other possible uses of the framework as engine for higher applications, such as dictionary searching or even for automatic creation of sign description databases from SL videos.

Further analysis will also target the building blocks of the language, by changing the semantic definitions of $\mathrm{PDL}_{\mathsf{SL}}$ operators to better suit SL. Changes to its syntax are also to be expected, in an effort to ease the development of extensions for different trackers and simplify descriptions. Finally, we want to steer further into 3D representation and the inclusion of non-manual features, important stepping stones towards higher level language processing.

7. References

Aronoff, M., Meir, I., and Sandler, W. (2005). The paradox of sign language morphology. *Language*, 81(2):301.

Barendsen, H. B. E. (1994). Introduction to lambda calculus.

Bossard, B., Braffort, A., and Jardino, M. (2004). Some issues in sign language processing. In Camurri, A. and Volpe, G., editors, *Gesture-Based Communication in Human-Computer Interaction*, volume 2915 of *Lecture Notes in Computer Science*, pages 90–100. Springer Berlin Heidelberg.

Cooper, H., Holt, B., and Bowden, R. (2011). Sign language recognition. In Moeslund, T. B., Hilton, A.,

Krüger, V., and Sigal, L., editors, *Visual Analysis of Humans*, pages 539–562. Springer London.

Curiel, A. and Collet, C. (2013). Sign language lexical recognition with propositional dynamic logic. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 328–333, Sofia, Bulgaria, August. Association for Computational Linguistics.

Dalle, P. (2006). High level models for sign language analysis by a vision system. In *Workshop on the Representation and Processing of Sign Language: Lexicographic Matters and Didactic Scenarios (LREC), Italy, ELDA*, page 1720.

DictaSign. (2012). http://www.dictasign.eu.

Dreuw, P., Stein, D., Deselaers, T., Rybach, D., Zahedi, M., Bungeroth, J., and Ney, H. (2008). Spoken language processing techniques for sign language recognition and translation. *Technology and Disability*, 20(2):121–133.

Dubot, R. and Collet, C. (2012). Improvements of the distributed architecture for assisted annotation of video corpora. In *5th Workshop on the Representation and Processing of Sign Languages: Interactions between Corpus and Lexicon.*, pages 27–30. Language Resources and Evaluation (LREC).

Filhol, M. (2008). *Modèle descriptif des signes pour un traitement automatique des langues des signes*. Ph.D. thesis, Université Paris-sud (Paris 11).

Filhol, M. (2009). Zebedee: a lexical description model for sign language synthesis. Internal, LIMSI.

Fischer, M. J. and Ladner, R. E. (1979). Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18(2):194–211, April.

Gonzalez, M. and Collet, C. (2011). Robust body parts tracking using particle filter and dynamic template. In *2011 18th IEEE International Conference on Image Processing (ICIP)*, pages 529 –532, September.

Gonzalez, M. and Collet, C. (2012). Sign segmentation using dynamics and hand configuration for semi-automatic annotation of sign language corpora. In Efthimiou, E., Kouroupetroglou, G., and Fotinea, S.-E., editors, *Gesture and Sign Language in Human-Computer Interaction and Embodied Communication*, number 7206 in Lecture Notes in Computer Science, pages 204–215. Springer Berlin Heidelberg, January.

Kervajan, L., Neef, E. G. D., and Véronis, J. (2006). French sign language processing: Verb agreement. In *Gesture in Human-Computer Interaction and Simulation*, number 3881 in Lecture Notes in Computer Science, pages 53–56. Springer Berlin Heidelberg, January.

Liddell, S. K. and Johnson, R. E. (1989). *American sign language: The phonological base*. Gallaudet University Press, Washington. DC.

Losson, O. and Vannobel, J.-M. (1998). Sign language formal description and synthesis. *INT.JOURNAL OF VIRTUAL REALITY*, 3:27—34.

Meir, I., Padden, C., Aronoff, M., and Sandler, W. (2006). Re-thinking sign language verb classes: the body as subject. In *Sign Languages: Spinning and Unraveling the Past, Present and Future. 9th Theoretical Issues in Sign Language Research Conference, Florianopolis, Brazil*, volume 382.

Microsoft. (2013). Tracking users with kinect skeletal tracking, http://msdn.microsoft.com/en-us/library/jj131025.aspx.

Ong, S. and Ranganath, S. (2005). Automatic sign language analysis: a survey and the future beyond lexical meaning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):873 – 891, June.

Wittmann, H. (1991). Classification linguistique des langues signées non vocalement. *Revue québécoise de linguistique théorique et appliquée*, 10(1):88.