Soundpainting language recognition

EPFL DH Master thesis

By  
Arthur Parmentier

# Introduction

To be written last.

# Table des matières

[I. Introduction 2](#_Toc41067149)

[II. Table des matières 2](#_Toc41067150)

[III. A brief history of Soundpainting 4](#_Toc41067151)

[A. Back in Woodstock 1974: emergence in emergency 4](#_Toc41067152)

[B. Developments 4](#_Toc41067153)

[1. A multidisciplinary language 4](#_Toc41067158)

[2. Fertility in Europe, worldwide spread in modern societies 4](#_Toc41067159)

[IV. Historical and theoretical context 5](#_Toc41067160)

[A. Signs for communication: a long history 5](#_Toc41067161)

[1. A very long time ago… 5](#_Toc41067162)

[2. Middle ages: neumes 5](#_Toc41067163)

[3. Creation of modern sign languages for deafs (?) 5](#_Toc41067165)

[4. Conduction & other contemporary forms of artistic sign languages 5](#_Toc41067167)

[B. Real-time composition/improvisation/generative music in the XXth century 5](#_Toc41067168)

[1. Cage & co 5](#_Toc41067169)

[2. Algorithmic music 5](#_Toc41067170)

[C. Linguistics 5](#_Toc41067171)

[Rousseau, Wittgenstein.. theory of signs: De Saussure (sign theory) 5](#_Toc41067172)

[1. Regular languages 5](#_Toc41067173)

[2. … link with generative music? 5](#_Toc41067174)

[V. (abstract) Mechanisms and concepts of Soundpainting (a theoretical model of SP) 6](#_Toc41067175)

[A. Preliminary remarks 6](#_Toc41067176)

[1. Context and scope of my personal observations 6](#_Toc41067177)

[1. The categorical and prototypical perception of concepts 6](#_Toc41067178)

[A. Mechanism 1: transformations (projections?) of concepts into signs on the physical space of the body 7](#_Toc41067179)

[1. Input space 7](#_Toc41067180)

[2. Transformations and output spaces 7](#_Toc41067181)

[Mechanism 2: 8](#_Toc41067182)

[B. Sign “overloading” 8](#_Toc41067183)

[1. Motivation 8](#_Toc41067184)

[2. Differentiation of signified across disciplines 8](#_Toc41067185)

[3. The difference between discipline (music, theater..) and technical apparatus, or “why we sometimes need to specify how to play a long tone, and sometimes not” 9](#_Toc41067186)

[4. Speculative theory of differentiation of signified across instruments (technical apparatus) of the same discipline 10](#_Toc41067187)

[5. Sign overloading 11](#_Toc41067188)

[C. Personal observations during SP practice 12](#_Toc41067189)

[D. Structural concepts (the grammar of the language, that results in its language) 14](#_Toc41067190)

[1. Identifiers 14](#_Toc41067191)

[2. Content 14](#_Toc41067192)

[3. Modifiers (content parameters) 14](#_Toc41067193)

[4. Mode 14](#_Toc41067194)

[Influence of context (from grammar to performance) 15](#_Toc41067195)

[VI. 15](#_Toc41067196)

[VII. Soundpainting recognition with Max/MSP 15](#_Toc41067198)

[A. A new configuration: motivations, goals, workflow & challenges 15](#_Toc41067199)

[1. Motivations 15](#_Toc41067200)

[2. Goals 16](#_Toc41067201)

[B. The big picture: general description of the system 16](#_Toc41067202)

[1. The choice of Max/MSP 19](#_Toc41067203)

[Description of each layer 19](#_Toc41067204)

[C. 19](#_Toc41067205)

[1. Part 1: Motion tracking inputs 19](#_Toc41067206)

[2. Part 2: Signs & dictionary management 25](#_Toc41067208)

[3. Part 3: Real-time classifier 28](#_Toc41067210)

[4. Part 4: Grammar parsing 30](#_Toc41067211)

[5. Part 5: Orchestra simulation 30](#_Toc41067212)

[D. Performance 30](#_Toc41067213)

[1. PoseNet and Wekinator settings 30](#_Toc41067214)

[2. Threading 31](#_Toc41067215)

[E. Learning in SP and numerical tool 31](#_Toc41067216)

[F. Potential & future of the tool 31](#_Toc41067217)

[1. Emotion recognition 31](#_Toc41067218)

[2. Classification problem with CRF 32](#_Toc41067220)

[3. The future 32](#_Toc41067221)

[VIII. Conclusion 32](#_Toc41067222)

# A brief history of Soundpainting

## Back in Woodstock 1974: emergence in emergency

Emergence of the first SP signs with the orchestra. Emergent in emergency (*Emergent* means beginning to arise and *emergency* means arising unexpectedly)!

<http://www.soundpainting.com/history/>

Origin of “SP” as a term

## Developments

### A multidisciplinary language

SP not limited to music performance. Signs started to be created by W himself from 90+ for multidisciplinary performance.

### Fertility in Europe, worldwide spread in modern societies

Walter gave his first SP workshop in Europe in the late 90’s and found a very fertile ground for Soundpainting in France, which now has probably the largest SP community over the world.

Today, Soundpainting is used for on every continent and several artists created their own signs for the needs of their own group or configuration.

Important point: W wants to make sure that the language stays normalized and universal. Think tanks, glossary/dictionary of signs, community relations…

Modern societies: to what extent can the SP concepts of performance, content, parameters, identifiers… apply to non-modern groups?

Transition: culture to concepts

Write more about the evolution, in order to introduce the concepts later. It should already give an overall broad picture of what SP looks like at this point

Hypothesis: historical link with neums (dev in fr)

# Historical and theoretical context

## Signs for communication: a long history

Chronological view (?)

### A very long time ago…

### Middle ages: neumes

### Creation of modern sign languages for deafs (?)

Lien avec Chant-signes IVT

### Conduction & other contemporary forms of artistic sign languages

## Real-time composition/improvisation/generative music in the XXth century

How to structure this part? By example? Topic?

Musique stochastique xenakis

### Cage & co

### Algorithmic music

## Linguistics

### Rousseau, Wittgenstein.. theory of signs: De Saussure (sign theory)

Emetteur/recepteur;;.

### Regular languages

### … link with generative music?

Check relevancy: Chomsky, hierarchical models, markov models, study of music within linguistic models.

# (abstract) Mechanisms and concepts of Soundpainting (a theoretical model of SP)

Now that we have seen a bit of the development of SP and the theoretical background it lies in, I would like to propose a model of Soundpainting that would explicit its construction mechanisms as a sign language, as well as the implicit operations that makes it an efficient language for art performance.

## Preliminary remarks

### Context and scope of my personal observations

The reader must be aware that my experience with Soundpainting and the observations that I consider in this section:

* Are rather limited in time (3 years span)
* Are very limited in terms of cultural diversity (most of my participating experience was in Lausanne, Switzerland or nearby (France) with performers I was familiar with, who mostly came from European music education institutions; at the exception of one experience in Rio de Janeiro, Brasil where I could both participate and observe SP practices)
* Are rather limited in terms of configuration (mostly groups of musicians, either in weekly sessions with 3-10 performers or workshops with guest soundpainter).

The reader is invited to compare my observation with his and criticize the models and interpretation I give in this section.

### The categorical and prototypical perception of concepts

The human (innate) categorical perception scheme has been studied in music REF and many other fields and plays a very important role in the construction of basic artistic concepts such as note, pitch, scale, line, hit… by constructing discrete categories out of a continuous set of elements.

We also know from research in psychology[[1]](#footnote-1) that a single concept can be modeled as a category of elements around a prototype, considered as the central point of the category. Moreover, people tend to define the concept itself by the characteristic traits of the prototype, whereas in general, it extends beyond such a definition.  
The prototypical scheme rejects the discrete notion of ‘limit’ or ‘border’, replaced by the continuous notions of graded membership (similarity to the prototype) and the fuzzy edges of concepts.[[2]](#footnote-2) But on top of the prototypical scheme, the categorical scheme introduces a rupture by either accepting or rejection an element inside the category based on its similarity with the prototype[[3]](#footnote-3).

*Let’s take the examples of birds… (does it needs an illustration?)*

## Mechanism 1: transformations (projections?) of concepts into signs on the physical space of the body

Key idea: forming a sign/gestural language means creating a mapping between concepts from different sources (fields?) and the physical space of the human person who signs. How is this done in SP?

OVERALL TRANSFORMATION SCHEME (illustrate the scheme of the transformation process: source concepts to signs that represents them, gestures that evoke them…)

### Input space

We can identify several repertoires (sources) of concepts in the input space of this transformation scheme

#### Concepts from artistic disciplines

Long Tone, minimalism…

#### Concepts from oral languages

Logical elements

#### “High/low” cultural representations of quiet/loud (volume), slow/fast (tempo), pitch…

Even though the concepts of volume, tempo or pitch may be universal, their mapping onto a low/high axis is defined culturally in modern societies.

#### …

### Transformations and output spaces

Sign = (symbol, icon, indice).

#### Creation of a symbol

… Describe here more about the process of creating a sign (examples, general rules…)

#### Creation of a icon

#### Creation of an indices

## Mechanism 2: Sign “overloading”

### Motivation

We have seen from the evolution of SP that a single sign can be use to signify a content[[4]](#footnote-4) not only for different instruments of the same discipline (1) but also across discipline (2). This what I call the “overloading” of a sign.[[5]](#footnote-5)

We will try to identify whether the operability of a sign in (1) and (2) involves mechanisms of different nature.

### Differentiation of signified across disciplines

We very commonly observe signs made of one signifier and several signified in oral languages and our everyday life. The signified is understood by an operation of “disambiguation” that depends on the context of the communication. In this section, we will see that a multi-disciplinary sign in SP can indeed have several signified that allow it to operate in several disciplines.[[6]](#footnote-6) Moreover, we will discuss the model of a bijection between signified concepts and disciplines., Finally, we will discuss the construction process of these multi-disciplinary sign to show that their signified are linked by analogies.

#### Existence of several signified concepts

To demonstrate the existence of several concepts (the signified) under a multi-disciplinary sign in SP, let’s take the common example of the LT, that we will use all over this section to demonstrate some of the mechanisms of SP.

For a musician, the LT is a concept preexisting to SP with a specific prototype, whose characteristic traits are “constant volume”, “constant pitch” (among others). But is the concept of the LT for a musician the same as the one for a dancer or a visual artist? To answer this question, let’s first remark that soundpainters often explain how to perform a LT differently for each discipline:

* “A fluid movement, without accent” for dancers
* “A freeze on the first syllable of a word” for actors
* “A note with constant volume and constant pitch over time” for most musicians
* …

They also often illustrate those descriptions with a prototypical example.

By looking at the description themselves, we can see that they involve different concepts: a “movement”, “roll” or a “syllable”, which cannot be considered equivalent. Moreover, we know from the history of SP that the concept of a LT was first borrowed from music and “extended” to other disciplines, i.e. that the multiplicity of signified of the sign “LT” is a voluntary construction[[7]](#footnote-7).

Although we illustrated the overloading mechanism with the sign “LT”, we can observe the same mechanism for other signs, for instance multi-disciplinary signs.

#### Bijection, surjection or injection between signified and disciplines?

Although we have shown the existence of several signified (several concepts) and their relationship with disciplines, one can wonder whether their relationship is a bijection (to a unique discipline corresponds a unique signified), surjection (each discipline has one or more signified) or injection (each discipline has either a unique signified, either no signified).

First, we can remark that there is a bijection in the case of the so called “multi-disciplinary signs” whose properties is exactly that each discipline has its own unique interpretation (signified) of the sign.

Then, we can also observe that some signs are very specific to a discipline (sometimes by construction), such that other discipline cannot interpret them. It is enough to conclude that in general, there is no surjection between signified and the set of disciplines.

Finally, on the question of injectivity, my hypothesis is that unlike oral languages, SP can be modeled as a regular language, hence a context-free language, such that the context (signs prior to the last executed sign) does not influence the meaning of the considered sign. We will see in a future section how this approach is successful in modeling the very basics of SP, while it is unclear whether all SP modes, rules and signs could indeed be represented by a regular language. In my hypothesis, there is indeed an injection between signified and disciplines, but I leave to experts in the domain the prospection of counter examples, such as context-sensitive examples that would demonstrate that a single sign can indeed have several signified for one discipline, that can be differentiated by the context instead.

### The difference between discipline (music, theater..) and technical apparatus, or “why we sometimes need to specify how to play a long tone, and sometimes not”

In the “Motivation” part, I differentiated the notion of “discipline” and “instrument”, that we could also call a “technical apparatus”. Indeed, one could wonder why the concept of a LT would be different in each discipline, but still the same for all instruments of the discipline.

In this part, I wil try to show you that the operability of a sign across disciplines is constructed “by SP” with learned (cultural) relations of analogy between signified whereas the operability of a sign across instruments of the same discipline is in general a construction external to SP that involves the human perceptual scheme.

DIFFERENCE D APPAREIL TECHNIQUE MAIS MEME PROROTYPE VS DIFF DE DISCIPLINE DONC DE PROTOTYPE (MEME SI L APPAREIL TECHNIQUE EST LE MEME).

### Speculative theory of differentiation of signified across instruments (technical apparatus) of the same discipline

Wrap-up: what is a long tone?

By taking the simple example of the LT, we will see how the categorical and prototypical perception scheme plays an important role in the mechanisms of SP.

What is a long tone?   
It would be tempting to define the long tone for each discipline by giving a set of characteristics that all long tone must have, for instance:

* In music, a constant pitch and volume over time
* In dance, a movement without accent…

We know however from the research on our perception scheme that the LT is not (and perhaps, cannot) be defined with such characteristics or traits. It shows us that the LT is not a musical element defined by a finite number of properties but is rather a category constructed by the human perception scheme[[8]](#footnote-10) around a “prototypical” element which exhibits the features that are usually said to define the concepts of a LT.  
In this framework, a content is perceived as “more” or “less” a LT, rather than either a LT or not.

### Sign overloading

I will try to show that that in general, the grouping relations linking those concepts together under a single sign are made possible by cultural schemes (analogies) and innate perception schemes of human mind. I will use the example of the sign LT to illustrate how these schemes allow for translating the sign LT into the concepts that are relevant for each discipline, instrument (or technical apparatus).

#### Innate perception schemes

##### How to perform a LT on percussions?

I often see questions arising about how to perform a sign for a specific instrument or discipline, for instance a percussionist asking how he should perform a long tone with drums.  
While the experienced performer will probably use different possible techniques intuitively (a fast roll, using brushes, playing on cymbals that have a long acoustic response), we can derive from this simple observation that the concept of the long tone doesn’t necessarily have a trivial interpretation and its realization may not be accessible for instruments like percussions.  
Let’s break down the conceptual operations that allow the percussionist to respond to a LT when its prototype is out the set of possibilities offered by his instrument.  
To achieve this, the percussionist would typically translate the concept of LT to another concept, the roll, easily accessible to the performer. The roll will allow him to approach the prototype of the LT by increasing the speed of the roll as much as he can.  
Another way to phrase this conceptually is to say that in the space of musical concepts, the prototype of the LT is the asymptotical, limit point of the concept of roll when its speed goes to infinity.

##### An implicit translation by our perception scheme

The important point illustrated here is that SP involves operations of translation from concepts whose prototype[[9]](#footnote-11) does not exist in a discipline to a concept of the discipline that relates to the prototype. In previous example, we saw that the percussionist could translate the sign LT to the concept of roll that is relevant to its instrument. But what is the implicit scheme allowing for this translation?

I interpreted the fast roll as an “approach” to the prototype of a LT in the space of musical concepts, suggesting that there is a metric that allow us to measure the distance between the roll at a certain speed, volume… and the prototype of a LT. Even though this topic is out of my field of expertise and of the scope of this text, we can remark that the acoustic response of a fast roll “approaches” the performer’s expectation of the acoustic response of a LT, such that the human perception scheme will associate the fast roll with the prototype of the LT.

##### Learning the translation

While the operations of translation are in general not obvious (hence the questions on the subject), I observed that most Soundpainters explicit what operations are valid to beginners in SP.  
  
IMITATION et mimetisme

SEul vs a plusieurs

#### Cultural analogies

## Personal observations during SP practice

My theory is that our categorical and prototypical perception scheme plays an important role on both how performers and soundpainters can interpret the artistic material that is being produced by the group, and how they will respond to the requests, i.e. what content the performer will produce.

I will try to support this theory by analyzing examples from my own experience as a performer, a soundpainter and discussions I had with other performers on the topic of the LT which appears as an easy example to observe some of the underlying schemes of perception and associations of musical concepts.

All 5 experienced soundpainters I have worked with teach the concept of a long tone by giving both characteristic traits and prototypical examples of a LT, showing “how it’s done” in each discipline. Their approach is usually to start defining orally the most characteristically trait of the concept, for instance “a fluid movement without accent” for dancers and then give illustrations using their body, with different speeds for each example. For musician, they would for instance mention that a LT is a sound with constant frequency as first definition of the concept and give examples by singing the prototype of a LT at different frequencies.

* From the examples given by the soundpainter, unexperienced performers are usually able to internalize the prototype of the concept of a LT for their discipline and produce examples of their own at different frequencies/speeds. I observed that the examples produced by unexperienced performers at first are very often close to the prototype.  
  But as performers become more experienced, I observed that they tend to increase the span of produced examples not only by broadening the distribution to the parameters that have been introduced to (volume, frequency, timbre…) but also by exploring different “dimensions” of the sound, such as vibrato, micro-distortions, extended techniques etc. In other words, they progressively “detach” from the prototypes of the LT concepts[[10]](#footnote-12) and their characteristic features by exploring more features of the content and reaching more extremal points. In other words, if we take the N-dimension space (one dimension per parameter of the LT) of all LTs accessible to a performer, whose boundaries are determined by the technical and imaginative limitations of the performer himself, the distribution of the production of a learning performer should first span a limited volume around the centroid of the space that can be considered the prototype of the concept of a LT. Then, the volume covered by the distribution and its variance would increase with experience and artistic research.  
  This expansion process was discussed especially in my experience with a year-lasting Soundpainting group, in which we had sessions dedicated to explore new generative processes and dimensions for LT and other very prototyped concepts. From the discussion, it was clear that our production of LT was largely prototypical and that extending the range of production required dedicated work and one explanation that I remember was that it takes a lot more cognitive load to produce a LT far from the prototype than a LT close to it. Therefore, under the constraints of immediate play, it was hard to propose something original. I conceived this training as a way to reduce the cognitive cost of the production of less-prototypical LT, therefore bringing more diversity to the responses of the performers.  
  During one week or shorter workshops that mixed both beginners and experienced performers in Soundpainting, I observed that experienced performers were responding to requests with a wider variety than beginners in Soundpainting as one could expect from previous observations, but also that performers with a greater technical level would also respond with a greater variety. My interpretation of this observation is that
  1. the cognitive load of the production of a content depends not only on the experience of the performer with SP, but also with his discipline
  2. the cognitive load is a key metric for understanding how far from a prototype a given performer can respond to a SP request

From my experience in Brazil, I can add to the latter the following remark:

* 1. the desire or willingness of a performer to respond to the request in a certain way is very cultural. In the previous discussion, I have interpreted the expansion of the variety of responses as a consequence of the decrease of its cognitive load for the performer, but it is important to remark that this expansion may not be observed at all if the performer himself is satisfied by a certain type of response, should it by prototypical or not. In fact, this expansion relies on the motivation of/relevancy for the performer to vary its responses for artistic reasons that depend on the context of the performance and the background of the performer, including his cultural background.[[11]](#footnote-13)
* Another consequence of the categorical nature of the concepts beneath SP signs is the inexistence of a clear frontier between the concepts themselves. For instance, one could argue that silence can be considered as an extremely low volume long tone, and purposefully respond to a request of LT with silence. My observation is that during learning phases, Soundpainters prefer that beginners show that they have understood the concepts by responding with prototypical examples instead of “extreme” examples.

*Goal: describe the concepts that SP deals with and propose a categorization of these concepts based on an interpretation of the mechanisms of SP (SP as a set of operations – transformations).*

## Structural concepts (the grammar of the language, that results in its language)

Those concepts are used to describe the structure of the language: its grammar. They are emergent from SP, in the sense that they have been formalized dozens of years after the creation of SP and refer to its particular grammar rather than objects that are also found in other contexts. I would call the latter ones “borrowed” concepts.

One should really think of these concepts as an equivalent of the “noun”, “verb”, “adjective” … that are used to describe oral languages that modern societies are familiar with in their oral languages.

It is important to note that sign languages (among which, SP) do **not** use the same concepts as modern oral languages to describe their grammar.

### Identifiers

### Content

### Modifiers (content parameters)

### Mode

All these structural concepts (grammar) are not culturally dependent.

DRAFT of ideas

* “Traditional” Soundpainting vs “Alternative” Soundpainting: a question a configuration (soundpainting as a language but not a configuration) means that it is not necessarily linked to the frontal relation between a composer (human) and an orchestra (of human performers). My use of the word “traditional” refers to the transmission and definition of SP by Walter, with his own use & configuration of Soundpainting performances. There is no “modern” Soundpainting but rather an expanding field of research and experimentations in how the language and its concepts can extend to very different contexts and configurations. Develop with Brasilian version of SP & baila baila.
  + The concepts depend on the configuration. For instance, a “note” in music is very different from the concept (or its absence!) of “note” in visual arts.
  + SP early concepts were built for music performances and translated later for other disciplines. (Q: why do we even use this word “discipline” in arts?)  
    -> Concepts inside SP are “stolen” from the configuration/context it deals with.
  + SP acts as a kind of translation from the concepts of different practices to signs
  + It is therefore important to understand that to my point of view, there is not a fixed set of concepts that would be incorporated by the SP language
  + but rather a fluid, dynamic and varying grammar

# Influence of context (from grammar to performance)

# Soundpainting recognition with Max/MSP

In this section, I will introduce my Max/MSP Soundpainting recognition tool.

First, we will motivate the creation of such a tool, capable of recognizing several SP signs but also of creating new ones, as a composer would do.

Then, we will explore the structure of this tool in a top-bottom approach, in contrast with constructivist model of SP that I presented in previous sections, by discussing the global structure of the program and then taking a deeper look into the features and key objects of each layer.

Moreover, we will investigate basic performance mechanisms to optimize the speed and accuracy of the system.

Finally, we will discuss the use of this tool for learning SP in connection with the theoretical aspects discussed in the previous parts. We will look at some implications of the tool for the formalization of SP and hopefully, the future of the language.

## A new configuration: motivations, goals, workflow & challenges

### Motivations

We have seen previously that SP is used in different configurations and their role as a super-structure, defining its set of internal grammars (modes), the roles of emitter/receiver (soundpainter/performer) and the “default parameters” as context-depend rules for the performance.

By creating a SP recognition tool, I am proposing new configurations for SP, in which a computer can process SP signs and take the role of one or several performers at the same time, aside other human performers or not. At the difference of what is possible with human performers, the program itself can only recognize signs from one emitter at the time (the soundpainter) and cannot emit signs itself. Among the set of possible configurations, I would like to mention two of them that I had in mind when starting this project:

* In the first configuration, the orchestra is simulated by the computer running the SP recognition tool (and possibly other digital gear), without other human performers.
* In the second configuration, the computer takes the role of single performer inside an orchestra with other human performers.

In general, the choice and use of a specific configuration is motivated by the qualities (in terms of creative processes, special layout, communication rules…) it has for achieving a certain result in the performance. Let’s discuss the qualities of these two configurations:

* In the configuration one, such a tool can be used as a learning tool for individuals aside the collective approaches to learning SP. As for now, the tool only covers the basic structures of SP and would only be interesting for beginners in SP or soundpainters who cannot rehearse with a group.
* In both configurations, such a tool is also interesting even for more advanced soundpainters for exploring new areas of composition with artificial intelligence, machine learning and their complex generative processes.
* In the second configuration, the tool can be used as a controller with other digital elements that are already part of the performance and sometimes used by the human performers themselves: effect processors, amplification mechanisms, mixing devices, recording devices and much more. It is sometimes the case that such elements cannot be controlled real-time performance because of the complexity of their interface or controls, or that it requires performers to manipulate them. The interfacing possibilities offered by Max/MSP makes it an ideal choice for controlling these devices within the SP language directly.

Of course, there are other possible configurations in which people would want to use the tool, such as remote performances over internet, where the tool could act as an interpreter from video to words and simple parameters for reducing the bandwidth of the information to pass over internet.

### Goals

In the frame of my master thesis, I have chosen to focus on the use case of my tool as a learning tool in the first configuration (simulation of the whole SP orchestra).

In analogy with the constructivist model we discussed previously, we know that the learning tool must have at least the following features:

* A mechanism of sign & dictionary creation
* A mechanism of classification between different signs and parametrization of different positions in the body space
* One or several grammatical systems for parsing the sequence of signs and gestures (one grammar for each mode)
* Audio/visual feedback in response to the soundpainter’s requests

The goal of my project is to implement these features, focusing on only one mode (the default SP mode) and implementing an audio feedback by simulating a small orchestra.

## The big picture

### General overview of the system

These goals and the constructivist approach to SP that was presented in the previous parts are direct inspirations for building the recognition with several independent layers, that are represented in Figure 1.

Each layer has a particular function that the user should easily be able to interpret. Inside each layer are different processes and objects that the user interacts superficially with from the interface of the program. For transparency, I have put in white the parts that I built myself and in blue the parts that are either taken from other works[[12]](#footnote-14) or only slightly modified for my project.

At the interface level, all layers are implemented inside Max/MSP, a graphical programming environment designed by IRCAM and optimized for real-time applications. The user interface has a very basic look and design. The user can see the whole patcher in the main window and is also able to access specific functionalities of each layer by using tabs.

At the processing level, Max/MSP itself has three different threads that it uses for processing the data passing through its compiled objects. For these threads, Max guarantees the synchronicity/ordering of events. However, Max also interprets node.js code that is processed in external threads asynchronously to Max internal threads. We will discuss the consequences and implementation choices of this remark in future sections of the report.



Figure 1 Summary of the recognition program structure

### The choice of Max/MSP

Before starting a finer description of the mechanisms of the program, I would like to motivate the choice of Max/MSP as the main programming environment hosting the different layers and functionalities.

There are several options for building such a tool, each with their own pros and cons. In my case, I have considered Unity3D and PureData as the main alternatives to Max/MSP.

The advantage of Unity3D over Max/MSP(/Jitter) is that it is endorsed by a much larger community, it is free and has a better potential in terms of graphics.

The advantage of PureData over Max/MSP is essentially that is it free and open source.

However, I think that Max/MSP is a better choice than those software for the following reasons:

* Max/MSP is a high-level object-based programming language with already optimized pieces for music and real-time applications that are easy to use and assemble. It allows for scripting in JS and Node.js, which makes it a powerful host for a huge number of scripts and tools developed by the web community and its graphical interface allow newbies in coding to catch up easily with what is going on.
* Max/MSP comes from IRCAM and is used by a community of artists and musicians with interests very similar to my projects.
* Max/MSP has a nicer visual interface than PD, is maintained by a commercial company whereas PD has not been updated for years.
* PD connects badly to external pieces (node.js scripts, java…) that are critical for my project.
* The graphical programming interface of Max/MSP can be transformed into a user interface easily (for simple demos such as mine) … or very extensively, with complex GUI objects.
* Given my programming skills and the time constraints of the project, Max/MSP was a much faster approach than Unity3D.

Although being a commercial software, Max/MSP patchers can be compiled to a standalone program for Windows and Mac OS, allowing to share the program for free.

## Description of each layer

### Part 1: Motion tracking inputs

As humans, we are equipped with cognition and recognition systems that allow us to discern a wide variety of objects such as bodies in space and to build features to identify (classify) movements and gestural signs from those bodies.  
Computers, however, are not natively equipped with such systems, so that it is necessary to build them in this project, according to the goals and objectives we defined previously.

The role of the motion tracking layer is to compute a set of motion features from the movement of the user. There exist several motion tracking systems with different technologies that allow computers to recognize bodies and compute several features that can be used to identify gestural signs.  
In SP, there are some body parts such as the hands that are much more frequently to sign than others, therefore they require more precise tracking than the latter to classify amongst the signs. However, all motion tracking systems available have a finite range of operation, i.e. they can only track motion at a certain scale. (just like the human cognition system).  
We can observe two main scales at which signs are performed: full body and hands. Although there are certain costly technologies that would allow us to deal with both scale in one model, I propose and discuss two different technologies that are each adapted and efficient for each one.

#### Full body scale: “skeleton” tracking

To construct our features for the identification of the signs, we must drop a lot of information from the input of the system, such as the webcam or motion tracking system we are using. For instance, information such as colors, certain body parts such as the belly or torso, sound, etc, are not crucial for the identification of SP signs. There may be cases in which there are, but they will not be covered by this project.   
Typically, the “skeleton” representation of the position of each broad body part of the soundpainter (hands, shoulders, head, hips, knees, feet…) would allow the computer to recognized most basic signs at the full body scale. Because of the structure of the body (articulations, rigid body parts…), only several key points are needed to model its skeleton. Then, the features that would allow us to classify different signs must typically reach a precision of the order of magnitude below the distance between two keypoints. Assuming that most body parts are separated by a distance of the order of 10cm, we know *a priori* that our motion tracking model at this scale must reach a precision of the order of the centimeter.

In the following part, I will introduce and motivate the choice of PoseNet as the main motion tracking model at the full body scale.

##### Introduction to PoseNet

PoseNet[[13]](#footnote-15) is a computer-vision model that can be used to estimate the pose of a person in an image or video by estimating where key body joints are in 2D space. Its performances on modern CPUs and GPUs allow it to run in real-time using a webcam or alternative low-latency video input devices.

In Max/MSP, I could adapt and upgrade two demonstration projects showing how to port PoseNet into Max with Node.js[[14]](#footnote-16) to build a simple user interface (1.2.) allowing the user to start PoseNet either in a separate window, within an Electron process or directly inside a Jweb object on the patcher. A demo of the process can be found in my [second](https://www.youtube.com/watch?v=jW6bo6XkhFo) and [third](https://www.youtube.com/watch?v=OmPFMT9mgOs) demo video of the tool. I have observed a slightly lower performance with the Jweb object than the Electron process on my computer and the Electron process has the advantage of having its dedicated window, allowing the user to move it, resize it and most important, to keep it visible while the model is running, otherwise PoseNet performance drops critically. Both hosts (Jweb and Electron) are perfectly inter-changeable in less than a minute (for loading the model).



Figure 2 View of PoseNet inside Max/MSP



Figure 3 Posenet skeleton tracking for "rest of the group", from the Electron external window

PoseNet allows the user to choose different models and internal parameters that will affect its performance:

* The architecture of the model (MobileNet or ResNet)
* The input resolution of the video input
* The output stride of the model
* The depths of the convolution operations (for MobileNet only)
* The size of the model (ResNet only, only affects loading time)

With these settings, the user can adapt the model to its hardware to get the best performance.

We will discuss the optimal performance settings in a next section.

##### PoseNet advantages

PoseNet has several advantages to its concurrent technologies:

* It takes it input from a webcam or any video input that can be recognized by the computer, so that
  + For many laptops with integrated webcam, there is no need of external hardware
  + It can be used with very common and cheap hardware in case the computer does not have its own webcam, making the costs typically very low
* It provides a direct feedback of its accuracy to the user by overlapping the skeleton joints with the video, allowing users to change settings according to how good they see the model performing
* It is an open-source project, led by giants (Google…) and supported by a vast community
* It is still under development and will probably continue to be improved over the years, so it has a much greater potential than hardware-dependent solutions that are getting obsolete very fast
* It integrates with Max (and other systems) very easily as it can be run in a little node.js server

##### The main shortcoming of PoseNet: depth

The only major shortcoming of PoseNet with respect to other motion tracking systems is that it only operates in 2D and does not model the depth of each body joint.

As a workaround, I first built a simple calibration process that allowed to compute the depth of the torso of the user as well as its angle to the camera. After some testing, I realized that it was useless in my use case, did not allow any better classification and would only bring noise in the data.

###### Depth in SP

In fact, there are also some specificities that allow us to recognize SP signs without any depth information in PoseNet.

My observation is that depth (z axis) is often not the most informative axis for recognizing SP gestures and even signs like “play” which uses the z dimension extensively can be recognized only by the movement of the body in 2D, from the point of view of the camera in front of the soundpainter, because there is little ambiguity with other signs. Depth would only be important in cases where two signs would be similar in 2D space and depth would be the only information allowing to classify them; which is rare in my experience of SP.

However, capturing depth is important for one special sign in SP, which is often called “entering the box” and almost only takes place in the depth dimension. Its specialty is that this sign, consisting in putting one foot in an abstract “box” in front of the soundpainter, is used in many SP modes to significate “execute the request now”; whether the request has been defined previously (default mode) or is being signed while the soundpainter has “entered the box”. It is only meaningful when executed simultaneously to other signs, for instance “go gestures” in default mode or contents in launch mode. The opposite sign “exiting the box” also has the very specific meaning of “getting back to default mode”.

How can we manage to get past that specific issue?

###### Recognition without depth: compromises and simplifications of SP grammar

Ideally, one would want to capture depth and abandon PoseNet for a better tracking method:

* Kinect systems can capture depth but have many other shortcomings in terms of performance, user experience and portability
* Motion capture suits (for instance, IR marker-based suits) are usually the most accurate and performant devices but their costs and and specificities make them unattractive for sharing the tool to the SP community
* OpenPose[[15]](#footnote-17) is the main realistic alternative to PoseNet at this moment; it supports 3D triangulation from multiple view (like two cameras orthogonal to the soundpainter) but could not be ported to Max without much hassle[[16]](#footnote-18)
* Some learnable triangulation methods are being developed recently[[17]](#footnote-19) yet far from portable to my project

PoseNet appears like the best compromise for this particular use case, its goals and under the constraints of this project.  
In theory, it would be sufficient to add a simple external hardware of software mechanism to know whether the soundpainter has “entered the box” or not to remove the greatest part of the problem. One could for instance think of using a numeric carpet of a simple tracker of relative feet distance on the z axis to achieve this.  
In fact, I chose to work primarily on the default mode as my use case. In this mode, the practical use of the “go gesture” “play” is always associated with “entering the box”[[18]](#footnote-20) while even if the use of other “go gestures” such as “Slowly enter” is not always with “entering the box”, a simplified version of the grammar can easily assume that “entering the box” is always performed with “go gestures”, without removing much of the SP performativity in this mode.

With this simplification in mind, a 2D tracking of the full body is enough to cover the default mode. But what about over modes, that also make use of the “box” in a different way?  
Interestingly, most other SP modes that I am aware of are using immediate requests rather than structured requests, and the soundpainter is always standing in the “box” when using these modes. In other words, signing “outside the box”, not in real-time, to make structured requests is particular to the default mode.[[19]](#footnote-21) By entering or exiting other modes (marked with their “enter” and “exit” signs), the program should be able to know whether requests must be considered immediate requests or structured, delayed ones[[20]](#footnote-22).

In conclusion, although more expensive or complex systems would allow for full 3D tracking of the body, PoseNet is suitable for building most features at the full body scale relevant to SP and is the most adapted technology in the frame of this project.  
From the observation of correlations between missing depth features and known signs/features, we are able to make a small simplification of the SP grammar by assuming that the requests must be executed immediately when “go gestures” are used, and that in every mode but the default one, the requests are immediate. Under these assumptions, PoseNet is sufficient to build all necessary features to recognize SP signs at the body scale.

##### Building features

Meaningful features to feed the classifier with need to satisfy the following properties:

* The set of features should reduce to the lowest number of dimensions, while still being meaningful, in order to avoid the so-called “curse of dimensionality” problem. It can for instance be solved with a principal component analysis which in general provides the best set of features to a given classification problem; however, in our case, we want the features to always remain interpretable to the user, so that we cannot afford such a transformation. Instead, we should only keep the most significant joints in PoseNet output such as the wrists or elbows positions, discarding less significant ones (for SP) such as the nose or hips. However, if the user wants to build its own set of signs that rely heavily on these body parts, he would have to use such features.
* The features must be invariant to transformations that are not meaningful and do not correspond to any sign. In particular, the features must be translation- & rotation-invariants and independent of the dimensions of the body of the soundpainter.  
  In practice, this is achieved with PoseNet by taking the X & Y distance between each joint and the nose, which is considered a fixed point and then normalizing the X dimension with respect to the inter-shoulders distance and the Y dimension with respect to the distance between the middle of the hips and the nose.  
  The feature invariance will allow a sign to be recognized independently of the soundpainter’s
  + location and orientation (as long as the soundpainter faces the camera with a relatively small angle and does not get outside its field of view)
  + body dimensions (assuming that the general proportions of the human body remain constant)

With these constraints in mind and from my previous knowledge of SP signs, I could first order the body joints in PoseNet by their importance for SP signs recognition for basic signs in default mode:

* wrists
* elbows
* shoulders
* all other joints

In fact, I decided to only use wrists and elbows positions, which gave the best performance in my initial tests, reducing the feature space dimension to 8 instead of 34 (all 17 body joints X and Y values).[[21]](#footnote-23)

#### Hands tracking

Similarly to the full body model, hands can be represented as a skeleton in which our features are built from the position of the hands in space. This time, a 3D model is necessary as several signs are ambiguous in 2D space and can only be classified by looking at whether the palm is facing the soundpainter or the opposite direction. It is for instance the case with the signs “two” and “volume”.

##### Hi5 Gloves

Just like for the full body model, I initially looked at several existing technologies to model 3D positions of the hands. At the time of my research (before early March 2020), there was no equivalent computer-vision model similar to PoseNet for the hands that I could integrate into Max.[[22]](#footnote-24)

Instead, a variety of dedicated hardware was already available on the market with three major technologies:

* Marker based gloves
* Flex sensors
* Inertial measurement units, typically using small magnetometers

The prices of this equipment is quite expensive on the market, from 1000 chf to 5000+ chf for most expensive models. The cheapest models suffer from important shortcomings:

* IMUs are reported to drift when magnetic fields are present around the gloves, preventing a close use of computers, cellphones and other electronic devices nearby.
* Flex sensors are only one dimensional. In the cheapest gloves, many important dimensions of the fingers’ movements such as the phalangeal joint angles or angles between two fingers are therefore not captured, whereas they are often used in SP and other sign languages.
* Some gloves are designed for specific software such as Unity and may not integrate with Max easily.

While most expensive gloves often provide solutions to these problems, I decided to start with the IMU based Hi5 gloves from Noitom, which is one of the cheapest one available on the market (1000 chf approx.) which provides a Unity and C++ SDK.

However, at the time of this report (early June), I have still been unable to test the gloves with Max. I have made two mistakes during the conducting of the project:

* Before buying the gloves, I had not realized that the gloves connection with the Vive trackers was not only a feature as they are marketed on their website and documentation but that the Vive trackers were also necessary for using the gloves directly with the provided Unity plugin. In fact, the connection with positional tracking devices was not explicitly documented as a requirement for the Unity Plugin in their documentation. We could therefore not use the gloves on the fly with the provided scene and needed additional scripting using the SDK.
* I had been late on testing the gloves and identifying that issue. Although it made sense to me to start by building all the core mechanisms of the tool and pipeline before integrating the gloves, it was a mistake not to test them earlier, so that I could have figured out this issue at early stages.

The EM+ lab offered me support to build a dedicated object in Max for receiving the gloves data. The first experiments with the C++ SDK were good but unfortunately, the difficult schedule of that semester did not yet allow us to go further.

##### HandPose

Around mid-March, Tensorflow released the new HandPose model[[23]](#footnote-25). I first heard about it from within the Max community, when a first wrap of HandPose into an Electron server (just like PoseNet) was shared on github late May[[24]](#footnote-26). In the following days, I contributed to designing an interface within Max that made the output of the model accessible to the user (Figure 4).

While I was stuck with the Hi5 gloves, HandPose provided me an easy and light way to test the multi-input pipeline and the combination of the two scales or recognition inside Max. An introduction to HandPose inside my recognition tool is shown in my [fourth demo video](https://www.youtube.com/watch?v=rKD5BMaHmI8).

HandPose properties are very similar to PoseNet, hence it runs in a similar Electron node.js server, also has several performance settings and joints that can be selected by the user as features for the classification model.

Contrary to PoseNet, HandPose models the hand in 3D. However, it cannot yet recognize two hands at the same time, although there are good reasons to believe that this will be implemented soon.[[25]](#footnote-27)



Figure 4 My re-worked HandPose port to max

In SP, signs that are made of hand poses in can have two versions: two-handed and one-handed versions. While HandPose only allows for using one hand at the same time, the soundpainter can therefore use only one-handed signs with the recognition tool. Another possibility would be to the split the video input from the camera for the left and right side of the body (only one hand in each side) and running the model on both sides. This has not been tested by myself by would theoretically work without problem if the user is able to do so.

Another important constraint when using HandPose is that the model only works as long as the hand is sufficiently close to the camera, typically within 2 meters from the camera. This greatly limits the ability of the soundpainter to use PoseNet and HandPose models on the same video source. While PoseNet has a greater accuracy when the full body is visible (not only arms), HandPose requires the user to typically get closer to the camera than the distance at which the whole body fits inside the camera’s view. I suggest using separate camera for each model, one that would be close to the soundpainter for HandPose and one further for PoseNet. I have not been however to test this configuration yet. If it is not possible, then I suggest making sure that the arms, torso and head are well visible for PoseNet, while leaving hips, knees and feet under the field of view of the camera and making sure that hands are put as close as possible to the camera when signing with hand poses.

#### Input manager

I initially created the ‘input manager’ to allow the user to select in the list of all possible inputs (PoseNet, Kinect, gloves…) those that he wanted to use and that the program should listen to. This way, each part of the system would know how many features to expect and how to route the data flow correctly.

Until late stages of the project, when I started working with both PoseNet and HandPose, it was unclear for me whether a single DTW model would be able to classify all signs from a sign input combining all the selected features from both HandPose and Posenet, or if I should instead use several models depending of the type of signs that the program should recognize (movement with the full body, hand poses, faders…). There was also no built-in way to make a generic input to model routing in Max, so that I was hesitating between building a fixed routing that would work for my use case but would not allow any flexibility to change the inputs to each model, or building the desired input manager routing matrix myself. I chose the second option which allowed me to build a very modular tool, in which I could add or remove models and inputs in the future following my needs and the constraints of the several SP modes.

The present input manager allows the user to add his own input to the system automatically, without manually creating additional routings all over the program, by following only a few conventions

* each input must send its data through a “send <input\_name>” object
* each input must also forward its “size” (number of dimensions of the input) before the recording is launched[[26]](#footnote-28) through a “send <input\_name>\_size” object

Then, on the input manager panel, the user can select what input should be used by each model. Because inputs have different data rates and dimensions, it is in general not possible to route more than one input to a single model. If two inputs are compatible (typically with similar data rates) the user simply can create a new input and merge the two original ones in a single one the way he wants to if he would like to route this new combination to a single model.

Une image contenant capture d’écran

Description générée automatiquement

Figure 5 Input to model routing interface. The routing matrix is automatically updated from the list of inputs and models that the user defines, allowing the user to add his own inputs or models in little time.

### Part 2: Signs & dictionary management

Once the inputs and models are defined in the program, the user can start recording signs and building its sign dictionaries for each model.

#### Sign creation

We have seen that creating new signs is one of the core mechanisms of SP. It is also one important feature of my program in which the user can either create his own signs or use pre-recorded signs, for instance that would have been created or recorded by other users.

In my program, a sign is defined two properties:

* Its name
* Its category, in analogy with the syntaxic model of SP : WHO, WHAT, HOW, WHEN, OFF, NEUTRAL & LOGIC [[27]](#footnote-29)

These properties are sufficient to allow the program to parse the sign, i.e. to construct a meaningful request from the temporal flow of signs.

The procedure for defining a new sign is shown in 2.1. (Figure 6).

Figure 6 User interface for defining new signs and recording them to the buffers of a model. First, the user can select which model he wants to use to recognize the signs he wants to add. Then, he must enter the signs and their corresponding category that he would like to record. Finally, he can set a few parameters before launching the automated recording session.

For the sign to effectively by identify, two steps are required after the sign has been defined:

* Record training examples
* Program the virtual instrument itself to interpret the sign

While the recording of training examples is an automated process that simply involves pushing one button, the programming of the virtual instrument or device that the sign should control is outside the scope of the program. For demo purposes, I have implemented myself the interpretation of some signs by Ableton Live and a simulation of an orchestra with the Bach Project, but the connection with other devices must be implemented later by the user himself.

#### Sign recording

The user can choose to either define one sign at the time and record one or several training examples for it, then saving the training data and adding another sign… or directly define a list of signs and recording all of them in the same session.

The recording session has the following form:

* 1. Initial preparation time of *I* seconds
  2. Each sign is recorded *N* times, in the following loop:
     + The recording is launched for *R* seconds
     + Break (preparation time for next recording) of *P* seconds

The user can change the values of *0<I, 0<N, 0<R<5[[28]](#footnote-30)* and 0<*P* according to his needs.

Each recording takes place in a different buffer of the Multiple Buffer (MuBu) objects (one Mubu object per model[[29]](#footnote-31)) and each active input data is saved into a different track. The user can navigate the recorded data in each buffer and track using the Multiple Buffer Interface (Imubu object).

Une image contenant capture d’écran

Description générée automatiquement

Figure 7 Imubu controls and interface.

After the recording session, the data contained in a given track of a given buffer corresponds to a (N+1) x L matrix with N being the number of dimensions of the corresponding input and L being the number of steps in the recording sequence (sampling rate x duration of the sequence). The additional dimension corresponds to the time-tagging of the data. There are two main reasons for time-tagging the data:

* When several inputs are recorded at the same time, each has its own output rate; time-tagging the data in each track guarantees that during playback, the rate of each track is preserved
* Individual inputs can have varying output rates over time, for instance PoseNet and HandPose that run on GPU. Although we will see that the Dynamic Time Warping classification does “warp” the sequence in time and is therefore not sensitive to small variations of data rate, it is safe to assume that keeping the data timing in place always would better represent the original movements of the soundpainter

Wrapping up, inside the MuBu object, a sign is represented by labeled multi-dimensional buffers that contain the motion tracking data corresponding to each recorded example of the sign.

Once the data has been recorded, the user is able to save the recorded buffers to files in the ./data folder, by using the dedicated button “Save buffers to file”.

#### Saving recordings to file: building a dictionary

What would be ideal would be to store the data in the following fashion:

./data  
 /track\_name (corresponds to the input name)  
 /sign\_label + unique\_id (.mubu or .txt)

The motivation for using this file and folder structure is that it best represents the data structure of the MuBu object itself and allows the user to clearly identify what the file corresponds to without looking at its metadata. The user would then be allowed to mix data from recording sessions that are inhomogeneous, i.e. with a different number of recorded inputs, by loading all the files that corresponds to the inputs he uses, even though they might come from very different sessions.

However, the MuBu object write and read mechanisms suffers from bugs[[30]](#footnote-32) that should be fixed by the developers in the near future (as of May 2020) and I had to implement a workaround before it gets fixed, by saving each buffer with all its tracks in a single file:

./data  
 /configuration\_#track\_name\_1\_#track\_name\_2…  
 /buffer\_name + unique\_id (.mubu, .txt)

This way, the buffer names are saved correctly, but the user is no longer able to mix data from different sets of inputs.

#### Loading pre-recorded signs

Loading buffer data from files is much simpler and can be achieved in 2.3. by a simple drag and drop of one or several data files in the dedicated zone (see Figure 6).

#### Summary

Wrapping up, the user flow of the sign & dictionary management layer is the following:

* If the user wants to record new signs, i.e. either record examples of a sign that was not recorded and saved previously or record more examples of a sign that was already saved into files, he must first define which signs he wants to record and then launch the recording session.  
  Once the signs are recorded in the buffer, he can save them to files by hitting the corresponding button if he is satisfied by the recordings.  
  If the user adds new signs again without saving the buffers first, the data that was contained in the MuBu object is lost.
* Once the recordings of new signs are finished, the user should load into the MuBu object the data files of all signs that he wants to recognize and classify, for all inputs that he would be using. This is done in by dragging and dropping the corresponding files from the data folder into the dedicated zone.

### Part 3: Real-time classifiers, regression or DTW models

Now that the user has been able to connect his motion tracking inputs and record a few signs, we will see how the system is able to “recognize” the signs in real time with one or several machine learning model.

#### A short introduction to cognition and machine learning models

The word “cognition” dates back to the 15th century, where it meant "thinking and awareness."[[31]](#footnote-33) The term comes from the Latin noun *cognitio* ('examination,' 'learning,' or 'knowledge'), derived from the verb *cognosco*, a compound of con ('with') and gnōscō ('know'). The latter half, gnōscō, itself is a cognate of a Greek verb, gi(g)nόsko (γι(γ)νώσκω, 'I know,' or 'perceive').[[32]](#footnote-34)   
Re-cognition can therefore be understood as the process of “another” cognition, that allows the identification of something already knew or already perceived.

One can wonder what the core processes of recognition are and try to reproduce them on a computer. Very common examples of computer-recognition come from the so-called computer vision field such as object recognition (classification) or detection convolutional neural networks (CNNs) whose performances are now close to that of humans. However, CNNs are trained on huge amount of data and are (in general) not models that can be interpreted by humans.

In the context of this master project and to offer the ability of creating new signs to the user, we must work with lightweight, interpretable models that can be trained fast and identify the signs that are performed in real time. In our case, the identification process is a simple classification process, in which we ask the classifier to predict the “class” of the motion sequence performed by the performer among a set of classes that have been previously learned by the model: SP signs.

We have seen that in SP, there are very different types of signs (movements, poses – at several scales). At the beginning of my project, I thought about recognizing all signs with a single Dynamic Time Warping model.

Indeed, two light-weight models are generally presented in the literature to classify time-sequences: Dynamic Time Warping (DTW) and Hidden Markov Models (HMM). In general, DTW is observed to be faster and more accurate that HMMs. [[33]](#footnote-35) Some works also propose combinations of HMM and DTW or modified DTW algorithms for gesture recognition.[[34]](#footnote-36)

My initial design was to sum up all the features that I would use (full body skeleton, hands skeleton…) into a single feature vector that I could feed to DTW.   
The first obstacle to this initial design was that all inputs would not have similar data rates, so that the combination of inputs would either result in a data rate equal to the slowest input or a very high data rate, equal to the sum of each input data rate, but with common values between two consecutive feature vectors. This situation would not be ideal as it would require more processing and would not represent accurately the movement.  
The second obstacle was that although DTW can recognize poses, it performs much slower than pose classifiers that are not time-dependent, such as SVM or decision trees. Ideally, one would want to construct one model per type of sign to be recognize.

I have chosen to implement two models for my final prototype: one for the full body with DTW and one for the hands with Adaboost decision trees.

#### Full body Wekinator DTW

Unfortunately, at the time of the project, I could not find any real-time implementation of DTW in Max/MSP.[[35]](#footnote-37)

However, the external software Wekinator[[36]](#footnote-38) offers a very efficient DTW implementation based on the FastDTW library[[37]](#footnote-39) with additional improvements for real-time performance and several internal parameters for its DTW model.

Max and Wekinator communicate in Open Sound Control (OSC). Although the user must launch Wekinator separately and perform basic operations on its GUI, most important parts of Wekinator can be controlled remotely via OSC, allowing Max to automatize certain operations, such its training process.

The “user guide” for using Wekinator with the project is the following:

1) Start Wekinator.

2) Set the listening port to match Max settings and click "start listening"

3) Set the OSC input address to /wek/inputs (default)

4) Change the number of inputs (#inputs) to match the size of your input in this patcher, as defined in the first layer. For instance, with PoseNet, there are two features per joint (X and Y coordinates) so #inputs = #joints\*2

5) Change the Wekinator output type to "All Dynamic Time Warping" with N gestures types, N equal or greater than the size of your dictionary of signs. It probably does not matter if you specify a greater amount of types, so you can also use any sufficiently large N if you do not know how many signs it should recognize; ultimately, Wekinator will simply never match the signs to those classes.

6) Set the ouput port to match Max/MSP settings and click next.

7) If any input is running, make sure that the “OSC In” indicator of Wekinator is green. If it is yellow instead, make sure you have some input running and try to open the view/OSC input status window and restart listening to the OSC. If it is red, check that the size of your input in Max matches the #input parameter of Wekinator.

8) You can now push the train button aside the Mubu object for the full body DTW model. The number of examples for each sign should show up in Wekinator.

9) Once the training is done after a few seconds, you can press the "run" button in Wekinator to start classifying your live input.

These operations may take 3 minutes at the first time use and less than 1 minute once the user would get acquainted to the process. Automatizing these steps would be very difficult from Max directly, as Wekinator builds its own file structures and I could not find a way to load a project in Wekinator from a command line directly. In the future, it is possible that the InteractML or RapidMax project will make the process fully automated.

Once the model is running, Max receives in real time the set of DTW distances from the real time sequence to each recorded sign sequence. By finding the minimal value in that set and comparing this value to a confidence threshold, we can find when a sign is being performed in real time.



Figure 8 DTW output GUI in Max. The middle black box are slides that show the confidence level of each sign in real time. Recognized signs are shown at the bottom.

We will address performance and accuracy aspects in a further section.

#### Hands Adaboost for decision tree

To recognize hand poses that are used in signs like “tempo” or “volume”, I chose to work with an Adaboost for decision tree classifier model in Wekinator. The configuration process in Wekinator is only slightly different and is fully detailed in my [fourth demo video](https://www.youtube.com/watch?v=rKD5BMaHmI8).

At the output of the model, Max receives the index of the most likely sign and can eventually threshold on the confidence of the classification to avoid false positives.

### Part 4: Grammar parsing

#### A finite state machine (FSM) or automata

From the models introduced in the previous section, we can recognize individual signs, forming a sequence in time, just like words form a phrase in oral languages.

The next step is therefore to implement the grammar of SP with a parsing mechanism that would then allow us to create requests or commands to each device that acts as an individual performer in the system.

We have seen previously that the default SP mode can be modeled as a regular language. This allow us to build a deterministic finite state machine (FSM) that can create meaningful requests from the time sequence of signs.

There are several ways to implement a FSM inside Max but the most convenient way that I found was using a node.js package called “Javascript state machine”[[38]](#footnote-40). By also using the Viz.js library[[39]](#footnote-41), I was able to display the FSM inside a Jweb object in Max as a connected graph that represents all the states and transitions of the FSM. This is a very nice visualization for learning the grammar of SP and also for programmers to have a direct feedback on their grammatical implementation, for instance when adding new modes to the tool in the future (see Figure 9).



Figure 9 Max SP automata (FSM) GUI

Yet, the FSM only implements the default SP mode and although they have a dedicated state, all logic elements are not implemented either, as they introduce a great level of complexity both in the automata and the interpretation by the device itself.

#### Forming a request to devices from a sequence of signs

For constructing the request messages, I took inspiration from the OSC protocol: at the output of the automata, requests are sent to the devices with the following recursive forms:

/device\_name/content/parameter value  
/device\_name/content/parameter/parameter\_of\_the\_parameter/…/… value1 value2 …

The way the request is formed inside the automata is by collecting each sign during the state transitions and assembling them into several hierarchical objects:

* The request object that stores each request in the following format:
  + At the first level, the index of the request
    - At the second level, the name of the devices
      * At the third level, the name of the contents
        + At the fourth level, the contents’ parameters

Une image contenant moniteur, écran, texte, intérieur

Description générée automatiquementAt deeper levels, the parameters of the parameters…

Figure 10 Representation of the request object inside Max

* The “distribution” object that stores what content each device is currently performing.
* The “reverse distribution” object that stores for each content, what device is playing it with what parameters.

For convenience, I also use intermediate objects that store specific types of signs (the “who”, “what” arrays) in the parsing process.

Finally, the “group” object defines and store all the groups of the performance and the “defaults” object stores default values for certain parameters. Both can be used to adapt the parsing to specific performance situation, in analogy with the default parameters and the conventional groups in SP.

Une image contenant capture d’écran, moniteur, écran, portable

Description générée automatiquement

Figure 11 Capture of the automata debug and convenience panels, where the user can take a look in each object internally use by the automata and send messages for testing or debug.

The details of the internal mechanisms of the automata and the operations it makes on those objects are not described in this report because of the complexity of their description in written English; they are available to the programmer by looking at the automata.js file that implements those mechanisms.

However, there are specific elements of the code that can bring interesting points to a conceptual analysis of SP categories.

#### Discussion of some implementation choices and representation of SP concepts and elements

##### “Play”, “slowly enter”: from the conceptual opposition between immediateness and delay to a continuous parametrization of time

To define “when” the events must happen in SP, it is very frequent to either use Play for immediate requests and Slowly for anything that needs to be delayed (and eventually precise the delay time using “within X seconds”). This way, there is a clear opposition between immediateness and delay, whereas for a computer, there are no such categories but rather a continuous parametrization of time, such that it is possible to synchronize events very precisely and define timing in ways much more complex than the immediate versus delayed dichotomy that is relevant for human performers.

I think that conceptually, using machines in SP brings new ways of thinking and working with time in SP. Because I am not an expert on that side of SP, I am not sure whether there are already signs for working with time at the precision and parametrization of machines (absolute timing, relative timing, continuous parametrization of time at the very low or very large scale…) but I have never experienced such signs. Of course, precision or parametrization is often not wanted in SP where the interest lies in the liberty that a performer has in his propositions. However, with machines, fine descriptions of time are often relevant and allow for a wide variety of results (think about signal processing, effects…). I find the idea of requesting a computer to delay a sequence by a few samples only or reaction synchronously to events a very powerful and interesting aspect of working with computer and hope that the language could evolve with the use of such technologies.

##### Content and modifier sharing the same state in the automata

Originally, I had separated the state “content” and “content modifier” in my automata, before observing that after the first request, they shared the same transitions and could be merged into a single state that I called “content modifiers”. This shows that the grammatical construction itself does not reflect the conceptual difference between each sign (even if the concepts of content and content modifier are relevant for human performers, they are not very different in terms of grammar).

In fact, it is possible to remark that the idea of a content always come with default parameters, such that signing a content is always equivalent to signing content + default or open parameters. On the other side, one could push the limits of the “content modifier” very far, such that one would observe that at the end, the content has totally changed. It shows that the frontiers or the concept of content and modifiers are very porous, just like we have seen in the theory of concepts previously. The fact that the content and modifier state coincide is to me a direct consequence from the porosity of these two concepts that can regroup under the single idea of “content modifier”, whether it is at a fine level (subtle changes in volume that are close for the human perception) or broader level (change from a long tone to a pointillism, that are very different conceptually).

##### The challenge of logic elements unveiling the contextual components of SP

Adding to the previous remark, we can see that using basic logic elements, we can change a “content” to a “modifier”, for instance in the sentence “whole group, movement with pointillism, slowly enter”. Pointillism is often explained as a content in SP but it obvious in that case that it acts as a content modifier.   
It is an important problem for writing a dictionary of SP. We already know from other languages that parsing a sign is only possible with its surrounding elements, i.e. contextual elements. Although it is tempting to classify “pointillism” as a content in a dictionary, the use of logical elements in SP prevents the system from knowing whether it is indeed a content or a modifier before getting additional contextual information.

Take the following example:

* First request: “percussion 1, actor 1, dancer 1, minimalism, play”
* Second request: “numerics, long tone, play”
* Third request: “minimalism…”

At this point, the finite state machine can not determine whether minimalism is part of the broad identifier “minimalism group” or if it is a content for the identifier “numerics”.

The relevant grammatical rule that is acting is the following:

Content + “group” -> Identifier

It is a very important limitation of the modeling of SP as a regular language, that I have not solved in my tool yet. Several options for solving this problem in a future implementation are discussed in a further section.

In all cases that I have identified, it is the logical element that are able to change the category of a sign and therefore “break” the FSM representation validity. I hope that my work and these observations will serve as a base for deeper studies and linguistic modeling of SP grammar in its “regular” and “contextual” components, a process which to my knowledge has not been done so far.

### Part 5: Orchestra simulation

From the OSC commands created by the automata, there is an unlimited panel of tools and ways to create an orchestra.

#### Ableton Live

I originally started by building a connection with the performance software Ableton Live because of its wide spread use in the community of live performing artists (DJs, live performers) and its useful features for live music such as the use of scenes and or the launch of clips synchronized with a particular metric and tempo.

Just like many DAWs, Live can be hard to control externally and only offer a limited API to its internal mechanisms. Controlling Live externally typically requires scripting a server in python that must be installed manually. However, LiveOSC[[40]](#footnote-42) has been shared by the community to help in this process by creating server that listens to OSC map it to Live parameters.

In my [third demo video](https://www.youtube.com/watch?v=OmPFMT9mgOs), I showed how I could use custom gestures to launch and stop clips in Live, just like a simple DJ controller would. It required only a little effort to convert the conventions I use in my OSC commands to match the LiveOSC format. By synchronizing all the clips at the same tempo and metric, Live allowed me to create a fun demo with very simple controls that “sounds good” and can easily be modified and customized by the users.

Of course, the integration of SP commands could not be pushed very far into Live without more complex scripting by extending the LiveOSC API, or perhaps by using additional custom Max for Live scripts.   
In particular, there can only be a single tempo at a particular time in Live. This is also unfortunately one major assumption shared by many DAWs. Some extended research showed me that there would be some workarounds change the tempo of a single clip of scene from within their naming, which would make it a better tool than other DAWs in which multiple tempos are almost never found. But getting into those modifications in real time would have been to far from my main objectives and given that Ableton is proprietary software, I did not want to spend too much time exploring that possibility.

## Performance aspects

### PoseNet and Wekinator settings

As mentioned previously, PoseNet as different parameters that influence its performances. It is however not obvious where lies the best compromise between accuracy and the number of poses processed by seconds (FPS). Let’s take a look at how Wekinator processes its input to identify the best settings for PoseNet.

We know that by default, Wekinator’s DTW is downsampling the data rate to improve the DTW speed, such that the best compromise in performance and accuracy is to keep the number of FPS just below what Wekinator can handle without downsampling for a sequence of 2 seconds[[41]](#footnote-43). With Wekinator’s default settings (max sequence size = 10), the ideal number of FPS is 5[[42]](#footnote-44). On a fast computer, it would be worth to change Wekinator’s default max sequence length to 20-40 and run with around 10-20 FPS, which have proven to be more than enough for SP recognition or disable downsampling.

Another question that would yet still need to be addressed is whether using very different rates for PoseNet among the training dataset or between the training dataset and real-time data would badly affect the performances of Wekinator DTW or not. Although DTW warps the sequence and should be able to match two sequences with a different data rate, they are still constraints on how far the algorithm looks forward in time to find a point that may match a sequence in the training set and possibly on how much the sequence can be warped. These remarks are only speculative at the moment and are simply left here to warn the user and programmer about the possible limits of the FastDTW and its implementation in Wekinator that takes compromises in order to be fast enough for real-time applications.

It is also to be tested how does the performance of Wekinator changes with the number of signs to be recognized. In the scope of this project, no more than dozens of signs were tested but it is possible and likely that its performance (accuracy but also speed) drops with the increase of the number of signs. These points should be investigated for future improvements of the tools, especially for working with bigger sets of signs.

The qualitative improvement of the translation- & rotation-invariant transformation was also observed during the initial tests with those 8 features by moving in space and taking slightly different orientations to the camera. However, the improvement given by the normalization of the body joints with respect to body dimensions is still to be tested with more users.

### Threading

## Learning in SP and numerical tool

## Potential & future of the tool

### Emotion recognition

#### From PoseNet to building features for recognizing emotions in body gestures

Although we discussed the choice of meaningful features for recognizing SP signs within the default SP mode, I would like to discuss the motion descriptors library for Max/MSP “Modosc”[[43]](#footnote-45) and its potential use in other SP modes.

Let’s look at the SP mode “shapeline”. In this mode, all gestures and signs made by the soundpainter (except the “exit” sign of the mode) are interpreted in a figurative way by the performers, i.e. in an iconic or suggestive way rather than in a symbolic way. For instance, the soundpainter could use his facial expressions to convey emotional content or imitate the throwing of a virtual ball in the space and let the performers interpret (abstractly and freely) the dynamics of the scene.  
Interpretation in such a mode, as we have seen previously, involves particular cultural knowledges as well as knowledges about emotions, facial expressions… whether they are cultural or not.

The EyesWeb project[[44]](#footnote-46) proposes several “expressive cues” for analyzing body movement and gestures in relation with their emotional content and creating models of interaction between gestures and musical languages, in analogy to what the mode “shapeline” offers in SP. Some of these cues are features such as the contraction index, fluidity, curvature of a movement, jerk (first derivative of the acceleration), symmetry with respect to different points of the body, directivity etc.  
Although the authors of the this system are not directly demonstrating that these are the features the cognitive system uses for triggering emotions[[45]](#footnote-47) , the experiments and interactive performances they present suggest that these features do capture some sense of the emotions conveyed through gestures.

In Max, the “Modosc” library allows for computing of some of these cues. In future extensions of the project, it would be relevant to explore simple interpretations of emotional contents through gestures with music or visuals contents in the shapeline mode based on such features for PoseNet, even though more performant models would probably come from non-interpretable machine learning models in the near future.

#### FaceMesh and face emotions recognition

Tensorflow has just released in March 2020 the Facemesh package[[46]](#footnote-48) that infers approximate 3D facial surface geometry from an image or video stream and that can be ported to Max just as easily as PoseNet or Handpose models.

One use of this package for SP could be building a lightweight expression and emotion classifier that could also be used in SP modes such as the shapeline.

There are already several convincing attempts at recognizing facial emotions but Facemesh advantage is that it opens the way for fast, real-time emotion recognition from 3D mesh, hence independent of the user’s face color, dimensions, eyebrow shape, etc. Just like a normalized skeleton from PoseNet allows us to build a simple yet efficient model for recognizing SP signs without heavy training sets and models, the Facemesh could allow us to recognize emotions as facial signs… but also creating our own, new signs with the face.

### Classification problem with CRF

* Le CRF tient compte de la proba de chaque signe plus de la position dans la requête pour classifier le signe

### The future

# Conclusion

SIGN = something that stands for something else; signified & signifier  
3 != types of signs: symbols, icons and indices

No synonyms in SP

Are there homonyms? YESS

Meronymes… to check/discuss : sense relations

Sign overloading: what is an analogy?

Quantum-like theory of concepts to model human interaction (interference) in responding to SP signs? <https://www.frontiersin.org/articles/10.3389/fpsyg.2016.00418/full>

Modes and defaults: attempt to define particular grammars inside a “universal” one?

Enquete statistique sur poids cognitif/temps de réponse/complexité de la réponse/débutant/experts

“categorization is the name of the cognition game and analogy is the mechanism that drives it all”.

Making analogy = raising the similar features of two mental things

Analogy is responsible for concept’s expansion.

Hierarchy of concepts? At least a complex form of structuring; Link with researchs in DH ontologies. Nice example: german way of constructing words

Utilization des signes motivée par la representation a priori du concept/résultat du soundpainter

Représentation mentale qui precede la réponse du performer; intéressant pour W (tout le monde?) si la réponse n’est pas telle qu’attendue

On ne peut s’adresser qu’à une discipline qu’on connait un peu

Immobilté du soundpainter dans la config ordi et config traditionnelle walter => aucune “deformation” du contexte de la performance artistique  
Notion d’espace

Distinction SP et performer: discussion de l’évolution, prospections de configuration possible?

Nouvelles configurations: le so

W a deja supprimé signe? Pk? Ex “race” supprimée du dict de l’académie  
Pas de synonyme en SP? Interpretation du synonyme dans les langues avec contexts: ??? nuance contextuelle (cf video )

Language SP manipulé par le sp (context free grammar) VS musical language (context sensitive) by performer

Context: importance au dela de la (context-free grammar): attentes du compositeur, configuration, par défaut…

Mode: préciser le sens en SP

“mode”, “forme”

Glossaire (definition des mots/concepts utilizes)

Biblio, table des illustrations.

4 semaines:

Facilité:

- partie technologique

* Brief history of SP

INTERACTML and why I dropped Unity

Splitcam

1. Eleanor H. Rosch. Natural Categories. Cognitive Psychology 4, 328-350 (1973), introducing the “prototype theory” [↑](#footnote-ref-1)
2. For a concise explanation of the ideas behind the prototype theory, check https://www.youtube.com/watch?v=mff\_sPnz\_gs [↑](#footnote-ref-2)
3. Ref M.R. class on pitch categorisation. Emphasis on the complexity of the categorisation process (non linear wrt similarity) [↑](#footnote-ref-3)
4. Note that the whole discussion of this section is only relevant to the signs used to signify contents. [↑](#footnote-ref-4)
5. In reference to the concept of overloading in programming languages. [↑](#footnote-ref-5)
6. There may be other cases in which a sign has several signified, even inside one discipline, but we won’t discuss this possibility. [↑](#footnote-ref-6)
7. We will discuss the motivations of this construction later. [↑](#footnote-ref-7)
8. it can also be manipulated by cultural schemes, but that is not the point here. [↑](#footnote-ref-10)
9. or the whole concept itself, but as we already saw, defining the frontier of a concept is not always possible. [↑](#footnote-ref-11)
10. I use plural here because I am assuming from the beginning that there is not a single concept of a LT for all disciplines but rather a set, a group, a of concepts, some relevant for only one discipline, some relevant for several disciplines, under the sign « LT ». In fact, I am following Maurice Bloch by thinking of signs and words as decoupled structures, where the sign LT can be thought as an alias for several underlying concepts (a line in painting, a fluid movement in dance…) : this is exactly what I am calling the overloading of signs. [↑](#footnote-ref-12)
11. In my case, from my experience in Europe, producing a large variety of contents was personnally exciting and part of the interest I and others share for the technique ; therefore it was an important consideration that may not be shared at all by other groups with different interests and motivations to use SP. [↑](#footnote-ref-13)
12. External parts that are “shipped in” with the program are distributed under GPLv3 license (or less restrictive). [↑](#footnote-ref-14)
13. With TensorFlow: https://github.com/tensorflow/tfjs-models/tree/master/posenet [↑](#footnote-ref-15)
14.  Posenet Node For Max: <https://github.com/tejaswigowda/posenet-node-max>

     Posenet for dummies <https://github.com/billythemusical/n4m-posenet-for-dummies> and original N4M posenet <https://github.com/yuichkun/n4m-posenet> [↑](#footnote-ref-16)
15. https://github.com/CMU-Perceptual-Computing-Lab/openpose [↑](#footnote-ref-17)
16. The only realistic approach would have been to use the pytorch implementation of OpenPose (<https://github.com/Hzzone/pytorch-openpose>) and then try to run the python scripts in Max (Max does not interpret Python natively) with <https://github.com/grrrr/py>. It is very unlikely to work and could not be tested in the frame of my master thesis. [↑](#footnote-ref-18)
17. The most convincing project is https://saic-violet.github.io/learnable-triangulation/ [↑](#footnote-ref-19)
18. I had a theoretical discussion with Walter around the idea that in theory, one could use « play » without « entering the box » in requests such as « Strings LT Slowly enter Perc. LT Play WG After 30 seconds Off Play <enter the box> », which would mean that when the soundpainter « enters the box » :

    Strings slowly enter with a LT

    Percussions play immediately a LT

    Everyone stops after 30 seconds

    By not « entering the box » when signing the « Play » or « Slowly enter » gestures, it would allow everyone to schedule the 30 seconds count at the same time to perhaps stay synchronized with an external process. It would make sense to differenciate the timing properties of the « go gestures » (delay start by a certain amount) with « entering the box » which has for specific meaning « execute the request now » for certain configurations, such as when working with computers and time-sensitive systems. In practice, with human performers, the decorrelation between the two is either rare or never used within the default mode. [↑](#footnote-ref-20)
19. One could ask : « why then is it the case that the soundpainter always consider the virtual « box », even when the grammar of the mode implies that the request is always immediate ? I believe that it has to do with how SP has developed from the default mode to others and the pratical point of reference that the « box » is a common symbol for immediateness in all modes. [↑](#footnote-ref-21)
20. In practice, soundpainters tend to omit the « enter » and « exit » signs of modes that are not ambiguous, making things more difficult without depth tracking in case that « entering the box » is the only mark of the change of mode (for instance by using the launch mode directly with « entering/exiting the box » instead of signing Launch Mode and Tear). [↑](#footnote-ref-22)
21. I did not test the performance of this choice quantitatively, but only qualitatively by testing the change in performance that I could perceive with and without shoulders. It is obvious from the SP signs that I consider that aside shoulders, no other body joints could increase the performance of the recognition in real-time (curse of dimentionality problem). [↑](#footnote-ref-23)
22. Several hand pose estimations models are presented here : <https://xinghaochen.github.io/awesome-hand-pose-estimation/>. The open pose model also has 2D hand pose recognition but in each case, the models could not easily be ported to Max. [↑](#footnote-ref-24)
23. See Tensorflow blog, March 09, 2020 : https://blog.tensorflow.org/2020/03/face-and-hand-tracking-in-browser-with-mediapipe-and-tensorflowjs.html [↑](#footnote-ref-25)
24. https://github.com/lysdexic-audio/n4m-handpose [↑](#footnote-ref-26)
25. See the MediaPipe repository : https://github.com/google/mediapipe/blob/master/mediapipe/docs/multi\_hand\_tracking\_mobile\_gpu.md [↑](#footnote-ref-27)
26. The information flow at the opening of the patcher follows a specific order. If implemented at startup, the triggering of the input size should be launched with a « loadpercent 91 » object to garantee that it is caught by the input manager. Otherwise, the number can be sent anytime before the recording is launched. [↑](#footnote-ref-28)
27. Other categories such as MODE are not implemented yet, given that the program only focuses on the default mode of SP. [↑](#footnote-ref-29)
28. 5 seconds is the maximum sequence length that I allowed the recording buffer to store. Internally, it is a constraint from the MuBu object that stores the buffers and require a « maximum capacity » for each track, possibly for memory allocation issues. However, it is unrealistic that a sign does take more than 2 seconds to be executed. [↑](#footnote-ref-30)
29. In further releases of the tool, i twill be possible to only use one Mubu object for all models, but I could not reach this level of genericity and complexity inside the frame of my project. [↑](#footnote-ref-31)
30. https://discussion.forum.ircam.fr/t/mubu-write-to-file-bugs-suggested-improvement/21714/2 [↑](#footnote-ref-32)
31. Revlin, Russell. *Cognition: Theory and Practice*. [↑](#footnote-ref-33)
32. Liddell, Henry G., and Robert Scott. 1940. "γιγνώσκω." *A Greek-English Lexicon*, revised by H. S. Jones with R. McKenzie. Oxford: Clarendon Press, and Stefano, Franchi, and Francesco Bianchini. 2011. "On The Historical Dynamics Of Cognitive Science: A View From The Periphery." *In The Search for a Theory of Cognition: Early Mechanisms and New Ideas*. Amsterdam: Rodopi. p. XIV. [↑](#footnote-ref-34)
33. Raheja, Jagdish & Minhas, M. & Prashanth, D. & Shah, Tarang & Chaudhary, Ankit. (2015). *Robust gesture recognition using Kinect: A comparison between DTW and HMM*. Optik - International Journal for Light and Electron Optics.  
    Carmona J.M., Climent J. (2012) *A Performance Evaluation of HMM and DTW for Gesture Recognition*. In: Alvarez L., Mejail M., Gomez L., Jacobo J. (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. CIARP 2012. Lecture Notes in Computer Science, vol 7441. Springer, Berlin, Heidelberg [↑](#footnote-ref-35)
34. Hiyadi, Hajar & Ababsa, Fakhreddine & Montagne, Christophe & Bouyakhf, El Houssine & Regragui, Fakhita. (2016). *Combination of HMM and DTW for 3D Dynamic Gesture Recognition Using Depth Only*.  
    Choi, Hyo-rim & Kim, Tae-Yong. (2017). *Directional Dynamic Time Warping for Gesture Recognition*. 22-25. 10.1145/3145511.3145526. [↑](#footnote-ref-36)
35. The MuBu library has a DTW object that can be directly used on the buffers but I could not find whether it could or how to make it work with real-time data.

    Frédéric Bettens from UMons presented in 2009 the num.dtw object for Max and PD, but it can not longer be found over the web as annonced in its introductory paper *Real-time dtw-based gesture recognition external object for max/msp and puredata* in Proc. SMC ’09, 2009 30-35

    Another DTW object for Max has been built on the online-DTW library: *An Online Tempo Tracker for Automatic Accompaniment based on Audio-to-audio Alignment and Beat Tracking*, G. Burloiu. In Sound and Music Computing (SMC), 2016, but it is not designed to be used as a classifier.

    The RapidMax object (<https://github.com/samparkewolfe/RapidMax>) implements a part of the RapidLib on which is also based Wekinator, but has less functionalities and does not implement DTW yet. [↑](#footnote-ref-37)
36. http://www.Wekinator.org/ [↑](#footnote-ref-38)
37. See *FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space,* Stan Salvador and Philip in Intelligent Data Analysis 11(5):70-80 · January 2004 [↑](#footnote-ref-39)
38. See https://github.com/jakesgordon/javascript-state-machine/ [↑](#footnote-ref-40)
39. https://github.com/mdaines/viz.js [↑](#footnote-ref-41)
40. The version that I am using is the following : https://github.com/ideoforms/LiveOSC [↑](#footnote-ref-42)
41. 2 seconds is the default value of the recording sequence for each sign. The implementation details of Wekinator are discussed in the course *Machine learning for musicians and artists, Working with time* on Kadenze.com and visible on <https://github.com/fiebrink1/wekinator>. [↑](#footnote-ref-43)
42. We indeed have FPS\*sequence\_length = sequence size. [↑](#footnote-ref-44)
43. Modosc can be found at <https://github.com/motiondescriptors/modosc>, which relevant papers. [↑](#footnote-ref-45)
44. http://www.infomus.org/eyesweb\_eng.php [↑](#footnote-ref-46)
45. See for instance https://arxiv.org/pdf/1402.5047v1.pdf [↑](#footnote-ref-47)
46. https://blog.tensorflow.org/2020/03/face-and-hand-tracking-in-browser-with-mediapipe-and-tensorflowjs.html [↑](#footnote-ref-48)