Designing Movement Driven Audio Applications Using a Web-Based Interactive Machine Learning Toolkit

Benjamin Matuszewski
CICM/musidance EA1572, Université Paris 8
STMS Ircam-CNRS-Sorbonne Université
Paris, France
benjamin.matuszewski@ircam.fr

Joseph Larralde
STMS Ircam-CNRS-Sorbonne Université
Paris, France
joseph.larralde@ircam.fr

Frédéric Bevilacqua
STMS Ircam-CNRS-Sorbonne Université
Paris, France
frederic.bevilacqua@ircam.fr

ABSTRACT

This paper presents a web based toolkit for implementing Interactive Machine Learning (IML) dedicated to creative audio applications. The toolkit, composed of a main library and a template application, facilitates the creation of experiences on collective musical interactions with a strong emphasis on real-time movement processing and recognition.

At its lower level, the `mano-js` library proposes a user-friendly API built on top of existing libraries. The library is designed to assist developers and creative coders in the appropriation and usage of the Interactive Machine Learning concepts and workflow, as well as to simplify development of new applications. The library is open-source, based on web standards and released under the BSD-3-Clause Licence.

At its higher level, the toolkit proposes *Elements*, a template application designed towards non-developer users. The application specifically aims at providing a mean for researchers and designers to prototype new movement-based distributed Interactive Machine Learning scenarios. The application allows to create a new scenario by simply providing a JSON configuration file that defines the role and the abilities of each client. The application has been iteratively tested and developed in the context of several workshops.

1. INTRODUCTION

The use of mobiles devices (e.g. smartphones) for multimodal interactions is promising. The ubiquity of embedded motion sensors in such devices potentially allows for using gesture and body motions to significantly enhance standard interaction techniques. This could thus enable for the creation of novel interaction paradigms beyond the standard use of the multi-touch screen. Interestingly, the use of such motion capabilities is currently largely lacking in most applications, with few exceptions such as the 'shake' gestures and the use of basic orientation detection. The most promising applications can be found in games or creative applications (as recently proposed by the musical application SNAP -Reactable[1]). Nevertheless, movement-based interaction remains difficult to implement since designers and developers are currently missing simple and user-centered tools to implement gesture processing and recognition..

In last years, we developed and demonstrated a series of libraries and applications for collective musical interaction using the WebAudio API [13–15]. However, while our previous web applications are using the motion sensors capabilities of smartphones, the gestural vocabulary remained simple, mainly using 'slow rotation' and 'hit' gestures. In order to significantly enhance such gestural vocabularies, we developed a complete set of tools to implement gesture recognition in our framework, which we report here.

Our approach is based on Interactive Machine Learning (IML) [1, 8]. In such an approach, the gestural vocabulary (or elements) is defined by the users or the designers. We particularly intend to facilitate any training procedure, for example by permitting to record a single example for each gesture class, and by providing the users the possibility to 'instantaneously' test how the system actually recognize the learned gestures [3, 9, 16]. To allow for rapid cycles in designing the gestures, more examples can be added to extend or refine the behavior of the system. Importantly, the gestural vocabulary can also be designed in a participatory setting [4].

In this paper, we describe how we developed a web-based Interactive Machine Learning toolkit, that can be used to design and implement movement-based interaction with mobiles. First, we present how we adapted `XMM`, an open-source C++ library for gesture recognition[2], within our existing javascript framework dedicated to collective musical interactions using WebAudio. This integration led to a new javascript library called `mano-js`.

Second, we present the template application called *Elements*, that has been developed and tested in several workshops. As we will see, our toolkit offers an original solution for Interactive Machine Learning in general. Indeed, the web plaform brings the possibility to easily share data and gesture models, allowing for designing gestural interaction in participatory design setting.

2. RELATED WORKS

In this section, we will shortly present related works concerning Interactive Machine Learning [5]. We will particularly present its use in creative audio applications for gesture-based control. The term Interactive Machine Learning has been broadly used to describe several types of applications. Here, we refer it as a machine learning approach where the user or designer is involved interactively in the creation of the database, in the choice of the algorithms

[1] http://reactable.com/snap/

[2] Originally developed by Jules Françoise, https://github.com/Ircam-RnD/xmm

Web Audio Conference WAC-2018, September 19–21, 2018, Berlin, Germany.

and possibly in the manual refining of their exposed parameters [8]. Please note that the term User-centered Machine Learning has also been used in [12].

Concerning gesture recognition, several toolkits such as the Gesture Recognition Toolkit [11], standalone applications such as the Wekinator [7], or Max externals have been developed [2, 3, 10]. Most of these pieces of software, found very fruitful for creative audio applications [6] were not—at least in their original version—available as tools usable on the web.

While there is a large number of machine learning tools available in javascript (even for deep learning, e.g. TensorFlow.js), they are generally not designed as the aforementioned software that was especially developed for Interactive Machine Learning. For these reasons, several attempts were found recently to also expose these tools to the web [17]. As we will describe, the use of web technologies also enables to expand Interactive Machine Learning towards what we call *Collaborative* Interactive Machine Learning.

3. MANO-JS

3.1 Overview

The `mano-js` library provides a high level and user-friendly API for the implementation of movement driven audio applications using motion sensors such as Inertial Measurement Units (IMUs) with accelerometers and/or gyroscope. The library is based on web standard and written in the javascript programming language. The `mano-js` library allows for gesture recognition, using an Interactive Machine Learning approach, by enabling users to record their own gestures.

The library is designed to propose a generic API, based on the RapidMix API, that allows to further extend it with new inputs and machine learning algorithms. Furthermore, communications between components of the library are handled using a JSON format, the Rapid-Mix JSON format, specified for enabling interoperability between multiple preprocessing and interactive machine learning libraries. [3]

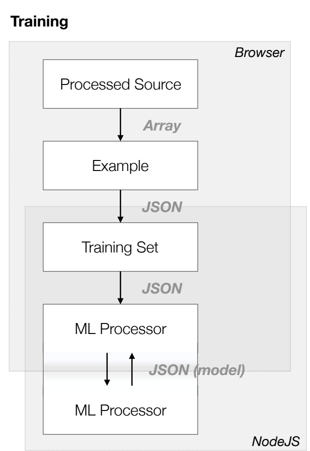

Figure 1: Possible data flows between different components of the `mano-js` library.

As shown in Figure 1, such a formalism enables communication between components over the network. It thus allows to create a wide range of application topologies, from simple and traditional client applications to complex distributed applications using shared models.

3.2 Implementation

As shown in Figure 2, `mano-js` is mainly built on top of two existing libraries. The pre-processing of the sensors data (accelerometers and gyroscope) is based on the `waves-lfo` library [13]. The machine learning part is built on top of the `XMM` [10] library, that makes use of probabilistic models for motion recognition. In particular, it implements Gaussian Mixture Models (GMM) and Hierarchical Hidden Markov Models (HHMM).

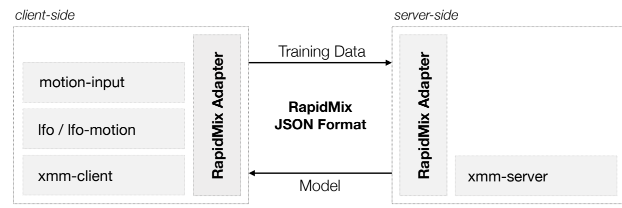

Figure 2: Underlying building blocks and formalization of the *mano-js* library.

The library implements a generic Interactive Machine Learning workflow. As such, it exposes of four classes dedicated to a specific task in the data flow and user workflow. Figure 3 shows a minimal example of the integration of the library and highlights how the four components interact with each others.

```
import * as mano from 'mano-js/client';
// instantiate classes
const processedSensors = new mano.ProcessedSensors();
const example = new mano.Example();
const trainingSet = new mano.TrainingSet();
const xmmProcessor = new mano.XmmProcesssor();
// create a labeled example and record data
// from the processedSensors
example.setLabel('my-label');
processedSensors.addListener(example.addElement);
// later...
// stop recording data from the processedSensors
processedSensors.removeListener(example.addElement);
// add the example the training set
const rapidMixJsonExample = example.toJSON();
trainingSet.addExample(rapidMixJsonExample);
// train the model
const rapidMixJsonTrainingSet = trainingSet.toJSON();
xmmProcessor
  .train(rapidMixJsonTrainingSet)
  .then(() => {
    // start decoding the processedSensors data
    // using the trained model
    processedSensors.addListener(data => {
      const results = xmmProcessor.run(data);
      console.log(results);
    });
  });
```

Figure 3: Data flow between the components exposed by the `mano-js` library.

`ProcessedSensors`

The `ProcessedSensors` class is responsible for acquiring the device's motion sensors data (accelerometers and gyroscope) as well

[3] The description of the format is available at https://www.doc.gold.ac.uk/eavi/rapidmixapi.com/index.php/documentation/json-documentation/

as for pre-processing acquired raw signals into higher-level motion descriptors. The abstraction is built on top of `waves-lfo`[4] [13] using an extension of the library dedicated to movement analysis.[5]

The class implements a generic listener API that allows to easily replace it to match other use cases.

`Example` *and* `TrainingSet`

The `Example` class can arbitrarily represent *timed* data, *multidimensional* data or *labelled* data. Once recorded, an example can be serialized and added to the `TrainingSet` using its RapidMix JSON representation. In a similar way, the `TrainingSet`—which acts as a collection of examples—can be exported to a specific JSON representation in order to feed the machine learning algorithm.

`XmmProcessor`

In the formalism proposed by the library, a *processor* mainly exposes two methods: `train()` that is responsible for creating a model from the JSON representation of a training set and `run()`, responsible to decode incoming data in real-time using the trained model. For now the library exposes one machine learning processor built on top of the `XMM` library, enabling the use of GMM and hierarchical HMM. [9, 10] Other algorithms will be added in the future to handle different use-cases.

4. ELEMENTS

4.1 Overview

Elements is a template application implemented on top of `mano-js` and designed toward non-developer users. The application is designed to provide an environment where users without programming knowledge can create their own instance of the application (e.g. behavior and mappings of different clients) by simply editing a JSON configuration file.

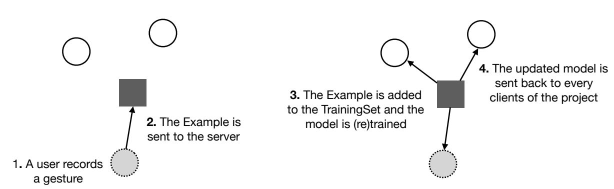

Figure 4: Usual workflow of `Elements`.

Each instance of *Elements* can host multiple *projects* in parallel. Figure 4 shows the generic workflow proposed by the application:

- A client records a new gesture example.
- The example is sent to the server to be added to the common training set.
- The model is updated according to the new training set.
- The new trained model is sent to all clients of the same project.
- Every client can make use of this new gesture.

[4] The *waves-lfo* library is available at https://github.com/wavesjs/waves-lfo

[5] The *lfo-motion* library is available at https://github.com/Ircam-RnD/lfo-motion

The centralized state of the project on the server thus enables the automatic sharing of the gesture models among clients of the project, allowing for creating a large set of collective interaction scenarios.

4.2 Players & Designers

A configuration file allows to define and configure multiple types of mobile clients in the application. For example, a user can define if a specific type of client can create new projects, record new gestures, update project and machine learning parameters.

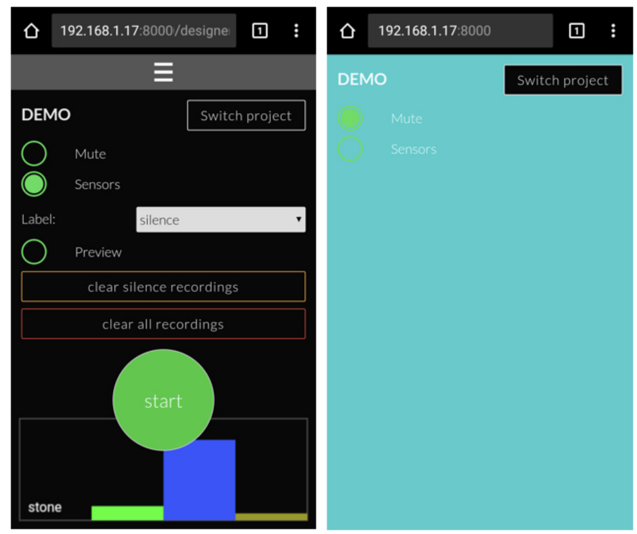

Figure 5: Examples of interfaces for different clients of Elements. Left: a client (called *designer*) exposing many control possibilities. Right: a client (called *player*) with control over simple options (e.g. mute).

Furthermore, as shown in figure 5, the graphical user interface can also be defined from the configuration file: from complex interfaces with many controls to very simple and minimalist interfaces without touch interactions. For example, a client that we typically refer to as *player*, generally exposes only one or two buttons (to control the mute and one other option). On the contrary, the *designer* allows for modifying all parameters of the recognition algorithms. Such granularity proved to be very useful in workshops situation where not every users must have the same level of control over the application. Also, it allows for modulating the degree of attention to the different parameters appearing on the screen.

4.3 Controller

The application exposes another type of client, the *controller*, that provides a centralized interface dedicated at controlling every aspects of the application as well as gathering feedback from users.

As shown in Figure 6, the interface provides information and controls at three different levels. At application level, for managing projects (e.g. creation, deletion, import and export); at the project level, for controlling machine learning parameters, or default state of all clients of the project; and finally at the client level for remotely controlling or monitoring a specific *player* or *designer*.

Indeed, the controller can display real-time visualizations of the processed sensors streams and decoding results as well as recreating the audio synthesis of each of the connected clients. This possibility proved to be very useful in many cases, from debugging to explaining how the system works in educational situations.

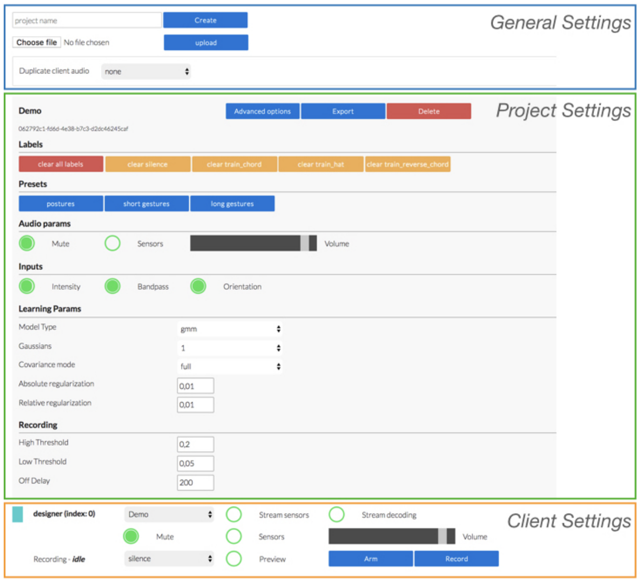

Figure 6: Elements' controller interface highlighting different possibilities of control.

5. CONCLUSIONS AND FUTURE WORKS

In this paper, we have presented a novel set of tools for the design and implementation of Interactive Machine Learning in the context of web-based collective musical systems. The proposed toolkit is composed of a javascript library, `mano-js`, that exposes a simple yet extendable API dedicated to developers and creative-coders, and of a template application easily configurable dedicated to non expert developer users such as researchers and designers.

In the current state of the toolkit, the proposed approach proved to be successful in several workshops.[6] We will pursue this development in providing alternative pre-processing and machine learning components. For efficiency and maintainability, we will also consider the WebAssembly format for these new components.

6. AKNOWLEDGEMENTS

The presented work has been developed in the framework of the Rapid-Mix Project from the European Union's Horizon 2020 research and innovation programme (H2020-ICT-2014-1, Project ID 644862). We would like to thank our project partners and our colleagues at IRCAM for their precious contributions to the project.

7. REFERENCES

[1] F. Bevilacqua, B. Zamborlin, A. Sypniewski, N. Schnell, F. Guedy, and N. Rasamimanana. Continuous realtime gesture following and recognition. In *Lecture Notes in Computer Science*, volume 5934, pages 73–84. Springer Verlag, 2010.

[2] J. Bullock and A. Momeni. Ml. lib: robust, cross-platform, open-source machine learning for max and pure data. In *Proceedings of the 2015 International Conference on New Interfaces for Musical Expression*, pages 265–270, 2015.

[3] B. Caramiaux, N. Montecchio, A. Tanaka, and F. Bevilacqua. Adaptive gesture recognition with variation estimation for interactive systems. *ACM Transactions on Interactive Intelligent Systems*, 4(4), 2015.

[6] http://gesturedesign.ircam.fr/

[4] A. Dubos, F. Bevilacqua, J. Larralde, J. Chevrier, and J.-F. Jego. Designing gestures for interactive systems: Towards multicultural perspectives. In *16th IFIP Conference on Human-Computer Interaction (INTERACT)*, number Part IV, pages 524–526. Springer International Publishing, 2017.

[5] J. A. Fails and D. R. Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003.

[6] R. Fiebrink and B. Caramiaux. The machine learning algorithm as creative musical tool. *arXiv preprint arXiv:1611.00379*, 2016.

[7] R. Fiebrink and P. R. Cook. The wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[8] R. Fiebrink, P. R. Cook, and D. Trueman. Human model evaluation in interactive supervised learning. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 147–156, New York, NY, USA, 2011.

[9] J. Françoise, N. Schnell, and F. Bevilacqua. A multimodal probabilistic model for gesture–based control of sound synthesis. In *Proceedings of the 21st ACM International Conference on Multimedia (MM'13)*, pages 705–708. ACM Press, 2013.

[10] J. Françoise, N. Schnell, R. Borghesi, and F. Bevilacqua. Probabilistic Models for Designing Motion and Sound Relationships. In *Proceedings of the 2014 International Conference on New Interfaces for Musical Expression*, pages 287–292, London, UK, United Kingdom, June 2014.

[11] N. E. Gillian and J. A. Paradiso. The gesture recognition toolkit. *Journal of Machine Learning Research*, 15(1):3483–3487, 2014.

[12] M. Gillies, R. Fiebrink, A. Tanaka, J. Garcia, F. Bevilacqua, A. Heloir, F. Nunnari, W. Mackay, S. Amershi, B. Lee, et al. Human-centred machine learning. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 3558–3565. ACM, 2016.

[13] B. Matuszewski and N. Schnell. LFO — A Graph-based Modular Approach to the Processing of Data Streams. In *Proceedings of the 3rd Web Audio Conference*, London, UK, 2017.

[14] S. Robaszkiewicz and N. Schnell. Soundworks – a playground for artists and developers to create collaborative mobile web performances. In *Proceedings of the 1rst Web Audio Conference*, Paris, France, 2015.

[15] N. Schnell, V. Saiz, K. Barkati, and S. Goldszmidt. Of Time Engines and Masters An API for Scheduling and Synchronizing the Generation and Playback of Event Sequences and Media Streams for the Web Audio API. In *1st Web Audio Conference. Paris*, Paris, France, 2015.

[16] B. Zamborlin, F. Bevilacqua, M. Gillies, and M. D'inverno. Fluid gesture interaction design: Applications of continuous recognition for the design of modern gestural interfaces. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(4):22, 2014.

[17] M. Zbyszyski, M. Grierson, and L. Fedden. Write once run anywhere revisited: machine learning and audio tools in the browser with C++ and emscripten. In *Proceedings of the 3rd Web Audio Conference*, page 6, London, UK, 2017.