Real-Time Sound Similarity Reconstruction by Features Extraction Analysis

Marco Matteo Markidis
nonoLab
mm.markidis@gmail.com

José Miguel Fernández
IRCAM
jose.miguel.fernandez@ircam.fr

ABSTRACT

This article describes a real-time audio analysis and resynthesis environment written for compositional research and for the development of music pieces for acoustic instruments and electronics. This work, called path∼, is implemented as an external for Pure Data[1] framework and it consists in an object that can be integrated in a Pd general real-time patch and used in combination with its internal and external objects. This environment is based on an ahead-of-time construction of a database and a related adjacency matrix through a features extraction analysis of chosen samples, on a real-time features extraction and concatenative synthesis upon a research of similarity nearest neighbors on the inner database. Although the present article treats different technical aspects, several applications and ideas on future developments are discussed in a perspective of realization of mixed contemporary pieces. In particular, we insist on the applicability of this environment for research and music composition.

1. INTRODUCTION

Sound reconstruction can be intended as a general class of synthesis tool. In its broader sense, granular synthesis can be thought as the first sound reconstruction application [1] . Over the last few years, various techniques have been discussed: in particular when we talk about sound reconstruction we mean concatenative synthesis and audio mosaicing. The main difference between these techniques and granular synthesis is that in concatenative synthesis the grains, which are usually shorts, are connected among themselves and they are related through a previous made analysis on chosen samples. This relation can be based on different methods and it is usually driven by the analysis: the goal is to set a criterion on which a best match can be found.

The basic idea of sound similarity reconstruction is to do a features extraction analysis on a sample and to set this criterion as a minimum distance between unit sample and the incoming sound. With the features extraction method, the similarities between sounds are underlined through the geometrical closeness in the feature space. This space is multidimensional and its dimensionality depends on the number and the type of the features used in the analysis. In order to perform a correct match a multidimensional database based on the ahead-of-time analysis is constructed. The research is conducted in this database and the distance is calculated between the incoming sound analysis, which is typically a multidimensional coordinate, and the database elements. In this contest, the term reconstruction refers to the resynthesis part of the method, in which the best match database element is played back through a granulation process.

The main motivation of this work is to develop an external inserted in the Pd world, so that a composer or a computer music designer can easily integrate it and develop patches for real-time audio composition. The aim of the project is not only to give a technical tool but also a compositional tool, intended as a more general framework for music production. Moreover, while in Max/MSP [2] background related projects and works were done, the lack of a similar object in Pd context was a great impulse [3] .

The main contribution of this work is the construction of an integrated environment within which all the processes described above are included in one object. This provides a more compact system that manages internally all the distinct steps. In addition, the internal design is thought optimized for real-time usage; several tasks can be performed ahead of time, for example in the loading-patch time: this lets the system to take care of real-time operations. Operations on big database, i.e. on big sample or a set of samples, can be executed with a reasonable latency and a study on this latency is given in the following. Finally, some additional facilities are present. In particular we create an asynchronous stream method and a multi-array usage that user can handle.

The paper is organized as follows: the next section describes the related works, making a distinction between software made for Max/MSP background and for Pd framework. The following part is about the methods used for path∼; in particular we focus our attention on the proposed algorithm and on the specific implementation with some reference to technical decisions taken about the code. Results, both technical and artistic, are presented in the fourth section. A major attention is given to the results that suggest us a further progress. Finally we will discuss the actual implementation and the future work.

[1] I. Xenakis' *Concret PH* is one of the first musical example.

[2] http://.cycling74.com

[3] This limitation is underlined even by William Brent in [2].

1.1 Ahead-of-time operations

In the present work, we always refer to a set of operations as done in ahead-of-time. Probably, this term doesn't represent exactly how path∼ treats this kind of operations in a behind-the-scene manner and the time that these operations can use. Indeed, the analysis method is thought to be done when the user initializes the patch. In this sense, the term ahead-of-time is correct because path∼ makes all the computations and the user can experience this performance. But other scenarios cannot be reconducted to this situation: for example during a rehearsal if the user wants to try different configurations he needs different analysis. Especially the add method breaks this acceptation, performing all the computations behind-the-scene, emphasizing the relaxation of real-time constraints. Being data analysis, the one-blocksize latency world is avoided. However the relaxation of this latency permits not to stop the audio flow, that is the primary intention of a real-time environment like Pd. In this sense the expression relaxed real-time used in [3, 4] can be useful and the idea that novel sound representations can be added is very intriguing for us.

2. RELATED WORK

After the pioneristic works of Iannis Xenakis, the attention of composers and researchers on granular synthesis and on granulation of sampled sound has grown. Nowadays, granular synthesis is one the most used synthesis technique in composition [4].
However, the general interest focused its attention on sound reconstruction from 2000s. Several projects were developed using these techniques. In particular, corpus-based concatenative synthesis was explored with CataRT[5].
CataRT system is a set of patches for Max/MSP using several extensions, like FTM, Gabor and MnM. The idea is to load sound descriptors from SDIF [5] files, containing MPEG-7 descriptors, or calculate descriptors on-the-fly, and create a descriptors space that uses short grains, called units, as points. The user can choose two descriptors and using implemented displays, like graphic canvas CataRT plots a 2-dimensional projection of units in this descriptors space, where the user can navigate and explore through the mouse position the descriptors space. Within this graphical environment, the expressive nature of the units can be underlined. The CataRT architecture can be divided in five parts, as follows: analysis, data, selection, synthesis and interface. Apart from the interface facilities that we have previously discussed, this system uses other integrated Max/MSP tools like Gabor library for synthesis; in addition FTM data structure were used for data management. CataRT was used for a piece for banjo and electronics by Sam Britton.
Another Max/MSP project was the realization of Marco Antonio Suárez Cifuentes' Caméléon Kaléidoscope[3], premiered at the Biennale Musiques en Scéne in Lyon in march 2010. For this production, a system combining FTM, Gabor, MnM and MuBu libraries was implemented. MuBu is a multi-track container representing multiple temporally aligned homogeneous data streams similar to data streams represented by the SDIF standard[6].
In Pd world, several works were proposed on timbre analysis and sound classification. Apart bonk∼[7] object, which is an onset detector object, a big project is the timbreID[2] library developed by William Brent. This library is a collection of externals for batch and real-time feature extractions. In particular, together with the presence of a set of low level descriptors, some high level descriptors were implemented, like mel frequency cepstrum and bark frequency cepstrum. The author shows that this set of high level descriptors has an higher matching compared to the low level ones[8]. Furthermore, the author shows that the use of multi-frame analysis compared to single-frame analysis and the use of combined descriptors, in particular a high level one plus low level features, increase the matching and decrease the ratio between the analysis' time and the score.
This peculiarity has a great interest in the sound similarity reconstruction, due to the possibility of a best recognition and therefore an improvement of the matching between the unit and the incoming sound. In addition the collection of single-descriptor external, a general object called timbreID is introduced, allowing the management of the information database.

3. METHODS

The path∼ architecture is presented in the following section. In the first subsection we talk about the structure focusing the attention on some peculiar methods and features; in the second subsection we explore in greater detail the implementation.

3.1 Algorithm

The general computation architecture is presented in Figure 1. As mentioned before, sound similarity reconstruction is corpus-driven. After loading samples, path∼ starts the features extraction and it creates a database. We chose a multi-descriptors strategy; in particular we follow a high level descriptor plus a set of low level features. path∼ has a mel frequency cepstrum descriptor, a spectral centroid one and a root mean square amplitude one. The computation of Mel Frequency Cepstral Coefficients requires a bank of overlapping triangular bandpass filters evenly spaced on the mel scale. A mel is the scale unit defined as:

$$Mel(f) = 2595 \times log_{10}(1 + \frac{f}{700}), \qquad (1)$$

where f is the frequency in Hz. Finally, in order to get the MFCCs we discrete cosine transform:

$$MFCC_i = \sum_{k=0}^{N-1} X_k cos[i(k + \frac{1}{2})\frac{\pi}{N}], \qquad (2)$$

[4] The idea of sound reconstruction is present in a great number of contemporary electronic pieces.
[5] Sound Description Interchange Format.

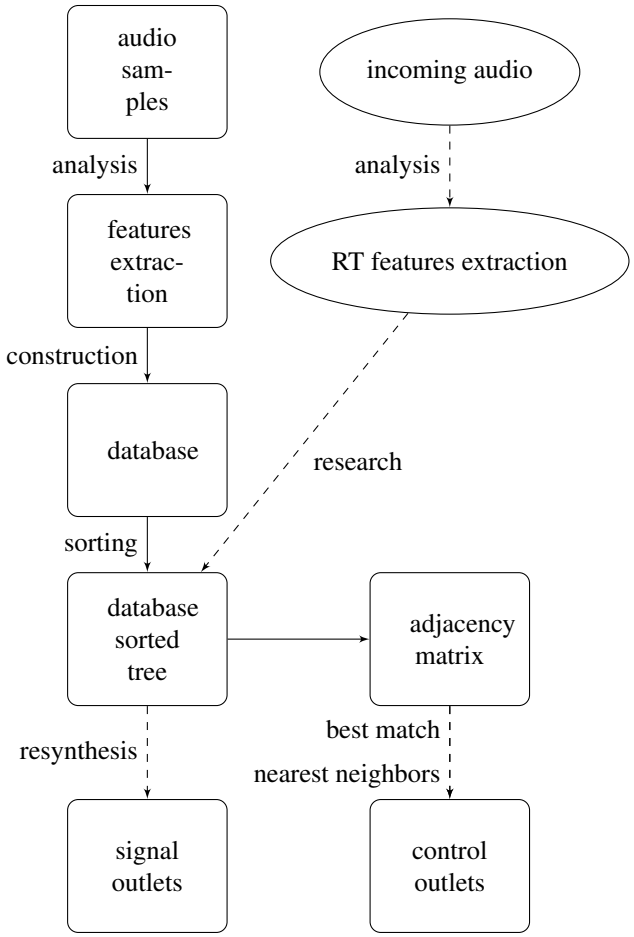


Figure 1: path∼ computation structure: ovals and dashed lines refer to real-time computations, rectangles and solid lines to ahead-of-time computations.

with $i = 0, 1, \ldots, M-1$. M is the number of cepstral coefficients, N the number of filters and X_k is the log power output of the k^{th} filter. The user can choose the frequency of the mel spacing during the creation-time. Varying this parameter, the number of resultant filters changes. This means that the dimensionality of our matrix will change respect to this parameter [6] . The default value of mel spacing is 250 Hz; the resultant dimensionality is 14, that is the number of MFCCs, plus 2, that is the dimensionality of the spectral centroid and of the amplitude descriptor, for a total of 16.
The spectral centroid is defined as the center of mass of magnitude spectrum.

$$SC = \frac{\sum_{k=0}^{N/2} f(k)|X(k)|}{\sum_{k=0}^{N/2} |X(k)|} \tag{3}$$

with $f(k)$ the frequency of the k^{th} bin.
The features extraction analysis is frame-based: the default value is 1024 samples. path∼ provides a Hann window for FFT analysis; the user doesn't complain with the creation and the passage of an external table. This operation creates a database, that is generally a matrix with the number of rows depending on the sample length and the number of columns fixed at number of MFCCs plus 2. Then this database is sorted in a binary tree. This sorting is useful in order to perform the ahead-of-time computation and to real-time research optimization. Every elements has a unique identificative index, that depends on the position in relation to the loaded samples. Moreover we create an adjacency matrix, in which for all the elements of database the indices of k-max-nearest neighbors are written [7] . The presence of this adjacency matrix permits to have a fast access to the pool of nearest neighbors, without an extra research.
A real-time operation is the audio input feature extraction, that gives a multidimensional point in the descriptor space. path∼ searches in the binary tree the closest coordinate and, given the direct access to adjacency matrix, outputs all control informations and creates a grain. This operation is sample accurate: the analysis is done according to the given user input and it is independent from the blocksize [8] . The control informations are the matched index founded in the database, the list with the desired k-nearest neighbors indices, the matrix length and the number of grain actually played. Apart from the sound content, the main characteristics of the created grain are the amplitude, the size, the envelope and the concatenation. These characteristics can be handled by the user directly with specific methods. The term concatenation refers to train length of the grain, i.e. the number of grains that follows the created grain. In this way grain trains can start creating sound textures beyond the single grain founded by features extraction matching.
Moreover we implemented a script interpreter thought to be a preset system. In this way, the user can write a script with a given syntax and in the creation-time path∼ loads it and creates a preset that can be called with a dedicated method in real-time. Optionally, the user can specify for every preset an identificative name. Apart from the interpreter, a simple debugger has been added which deletes from the presets the not found symbols and signals them to the user. Furthermore the complete preset list containing the debugged list with the various preset names can be viewed in a GUI editor that can be called clicking on path∼.

3.2 Implementation

path∼ architecture is based on that we have called Pds. Pds, namely Pure data structure, is a data structure that contains all the informations that we obtained from the chosen samples. In particular, it manages the nearest neighbors relation and the adjacency matrix apart the acquired samples. Every time the user loads samples, a Pds is created behind-the-scene and when this operation is completed, path∼ does an atomic compare-and-swap synchronization. In this way, a lock-free implementation is provided. Pds creation is done using pthread signal function; we can perform all computations in a dedicated thread and finally swap the entire data structure. This concurrent structure

[6] Just to understand the order of magnitude: 500 Hz = 6 filters, 250 Hz = 14 filters, 100 Hz = 38 filters.

[7] k-max is the maximum number of accessible nearest neighbors and it defines the adjacency matrix number of columns. Default value for k-max is 64. It can be modified at creation time.

[8] By default, 64 samples in Pd.

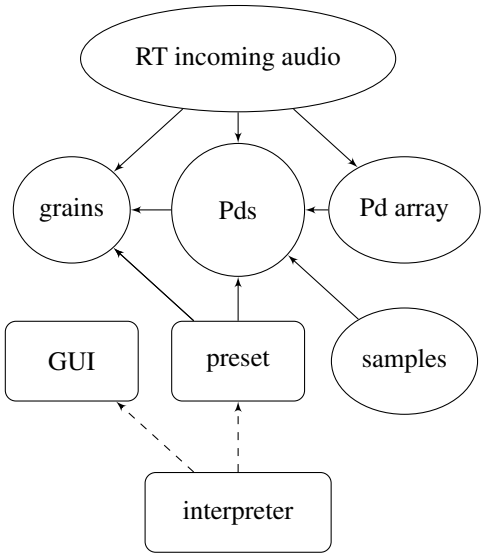


Figure 2: path∼ architecture. The rectangles and the dashed lines are the creation-time elements while the ovals and the solid lines stand for the changeable elements.

permits to add or remove in real-time samples or Pd array without breaking the audio flow.
In Figure 2 it's possible to see how the relation between different elements is implemented. We stress the complexity of real-time computation. Indeed, using our ahead-of-time technique the nearest neighbors research is a typical $\mathcal{O}(log(N))$ and the research in the adjacency matrix is $\mathcal{O}(1)$, there N are the rows of the sorted database.

4. RESULTS

In the following we present two different types of results, one technical and one artistic. In the technical report we made a performance results, focusing our attention on two kinds of tasks. In the artistic analysis, we conduct a report on the possibilities given by path∼ and we propose and stress several feasible application in mixed contemporary composition.

4.1 Performance results

Basically we are interested in two kinds of performance results. The first one is the latency introduced in the research of the nearest neighbors with respect to the change of the data dimension. The second one is the rate of matching with respect to the time analysis. We propose two tests that the user can replicate on his workstation. We perform these tests on a Mac BookPro Early 2011 13" 2.7 GHz and Pd Vanilla 0.46.6.

4.1.1 Latency

In order to perform this report, we implement a test method that takes as parameter the length of the database. This method constructs a random generate database on which all ahead-of-time computations are made. The statistics include essentially the time consumed with the real-time operation. It can be viewed through the latency method that stores these informations.

Rows \ Dim.	8	16	24	32	40
100	0.031	0.034	0.099	0.054	0.119
500	0.111	0.118	0.144	0.086	0.148
1000	0.138	0.193	0.157	0.151	0.137
5000	0.200	0.449	0.447	0.710	0.700
10000	0.407	1.049	0.949	1.037	1.380

Figure 3: Latency of real-time operations.The resultant latency is given in milliseconds.

In Figure 3 we show the latency relative to different measurements. Moreover in Figure 4 we have a 3D visualization of this table. Even for a database of dimension 10000x40 [9] a reasonable latency is reached. We made this test on a statistics of 100 bang. We don't provide an error: however we can see that small fluctuations are present, so these numbers can be useful as a general measurements. Finally, this test provides a good upper bound. In fact, the random numbers should be uncorrelated; this means that, on the contrary of usually analyzed incoming sound and samples that are strongly correlated, the possibilities of a best match are few increasing the numbers of visited coordinates in the sorted database.

4.1.2 Recognition

The test that we carried out about recognition is quite straightforward. We analyze a baritone saxophone recording of 37 seconds performed by Claude Delangle. This extract is taken from José Miguel Fernández's *Dispersion de trajectoires*. In particular, this piece makes a great use of contemporary extended techniques on saxophone, making it a good candidate for our purposes. In Figure 5 we show an extract of the score. The test consists in perform the ahead-of-time computation of the chosen sample and directly analyze it in real-time. If the algorithm works properly, we have to see in the match outlet a linear increasing number. As we expected, this is more a consistency test than a true test. We record the output in a text file using textfile object and we perform the analysis with a Octave script. Even for a 64 samples analysis, that means a database that has more than 25000 rows, with a dimensionality of 8, i.e. one the worst scenario, we get a 100% matching [10] . This result encourages us to start to try a true live test.

4.2 Application in mixed contemporary composition

The use of real-time analysis systems for instruments' sound is a key field in the interaction between acoustic instru-

[9] Just to understand the order of magnitude, a 10000 database length is almost 4 minutes of samples at default window analysis, i.e. 1024 samples.

[10] Just as a matter of curiosity, for this computation we obtain a database length of 25708. The latency of the real-time is the minimum reachable, due to the best match condition. In this case we have a latency of 0.032 ms, that is on the same order of 7000 rows 8 dimensionality database randomly constructed.

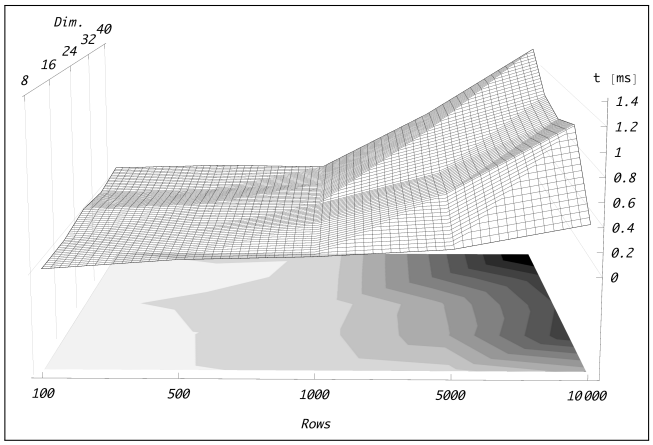


Figure 4: Latency of real-time operations. On x-axis we have rows, on y the dimensionality and on z axis the latency in milliseconds.

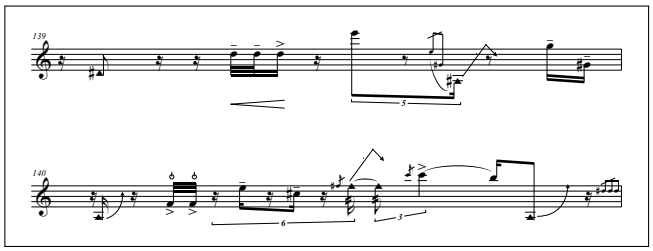


Figure 5: Extract of score of *Dispersion de trajectoires* that we used for our recognition test.

ments and electronics. It provides a system for generating musical structures in real-time based on the synthesis and sound treatments. While over the last few years various methods of analysis of the sound signal, e.g. audio features extraction, have been implemented these have not been exploited in all them possibilities and there are only a few musical works that use these methods in an organic shape with a musical and compositional thought.

The implementation of path∼ in Pd is just a tool that aims to bring the world of audio analysis and audio features extraction to create an environment for data-driven concatenative sound synthesis that can be adapted organically to musical and compositional discourse. path∼ possibilities are already numerous in this first version and give the opportunity to the composer or sound designer to experiment and to create different types of electroacoustic material in real-time or to generate electroacoustic sounds.

The ability to load and analyze various sound files in order to create a database as well as the ability to change dynamically the characteristics of grains amplitude, duration and envelope by a script-based preset system incorporated directly into the object gives to path∼ a great versatility and a flexible character. From the point of view of analysis and resynthesis we can highlight the threshold method that allows, depending on a given value, to trigger grains that are closer than this threshold value to the sound input. This method, together with the ability to load, analyze several amounts of samples and different types of sounds and select specifically with the list method which of the loaded sounds can be used, allows to create responses of concatenative synthesis extremely articulated and a close and coherent consistency to respect the real-time audio input. This possibility introduces a distinctive idea of distance that assumes a compositional aspect. In this sense the distance become a compositional parameter respect to a grain can be triggered or not, creating an asynchronous stream of grains. The idea of distance can be explored even with the weight method, that permits to change the relative weight in the distance calculation for every single dimensionality. This means that composer can affect one dimension to respect the others and create configuration in which one feature has major weight.

Moreover, a great extension to path∼ is given by the possibility to add in real-time a Pd array to the previously loaded and analyzed samples; in this way the analysis of a recorded on-the-fly sound is possible. This facility permits to record acoustic instruments directly during the rehearsals or the concert, leaving the composer free to add or remove musical segments and to include concatenative synthesis on live performed sounds.

Furthermore, path∼ uses a vectorized audio outlets control system. This system permits, during the creation-time, to specificate the number of outlets, i.e. the physical signal outputs, and the number of virtual channels, i.e. the grains number. The main motivation of this dynamical management is the control over different loudspeakers configuration, for example during the in-studio experimentation and the live performance, or in order to perform the same musical piece on different loudspeakers numbers and arrangement.

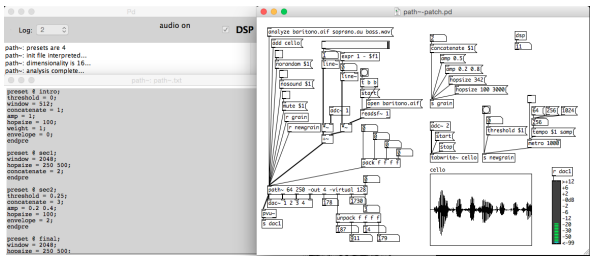
Finally, path∼ accepts a no-sound method that excludes from the computation the resynthesis part of the algorithm leaving access only to the control informations. In this way, applications without concatenative synthesis can be done saving CPU consume.

5. DISCUSSION AND CONCLUSIONS

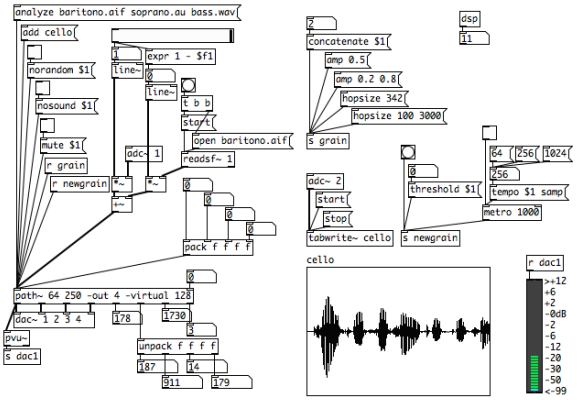
Even if this is only the first release of path∼, we have obtained several and encouraging results. In particular, the introduction of Pds opens the way to new directions although the multithread management and the construction of a concurrent structure are not trivial in Pd due to his nature. However, the good matching and the latency due to the applicability of this system even for large database indicate that the applicability in contemporary music is possible.

The idea of ahead-of-time operations permeates this project with its aestethics and its philosophy and puts us in the privileged condition that the algorithm inspires the musical composition and viceversa.

The primary aim of this work will be the production of a contemporary music piece for acoustic instruments and electronics. In order to obtain this, several improvements are planned. In particular a polyphonic version will be useful in an ensemble contest and a greater control on spatialization and on single grain control localization can be desirable. Moreover, time will be invested on the improvement of the GUI editor, giving to the user the possibility



(a) An application patch. On the left the GUI with various presets.



(b) Detail of Figure 6a. Various methods presented in subsection 4.2 have been used.

to change preset dynamically. Finally a growing interest will be dedicated to cluster analysis implementation and on improvement on recognition in short time analysis, particularly trying to develop some specific pattern recognition respect to sounds made by extended contemporary techniques.

Although the work is still in progress, the authors are using path∼ in their sound experiments. In particular, Marco Matteo Markidis is using it in his radical free improvisation duo Adiabatic Invariants. Actually, in *Cattedrali di Sabbia* a heavy use of path∼ has been done.

path∼ has been tested on OS X and Linux and on all Pd versions actually available today [11] , i.e. Vanilla stable release 0.46.6 and test release 0.47.0 and on Pd-l2ork alpha 0 release. path∼ is an open source software and it will be released under GPLv3. We plan to make it available via Deken, the externals wrangler for Pure Data, in the future stable version. Actually, in order to obtain this first release just contact Marco Matteo Markidis [12] .

6. ACKNOWLEDGEMENTS

We would like to thank Stefano Markidis for his suggestions and proposals; Martino Traversa and Fondazione Prometeo for the support and the encouragement in this period. Finally a special thank is to Daniele Pozzi without which our interest in this work would not be born and to Liarss in the person of Luca Gazzi, who dedicate to our project his patience and many hours of his time.

[11] June 2016.

[12] mm.markidis@gmail.com .

7. REFERENCES

[1] M. Puckette, "Pure data: another integrated computer music environment," in *Proceedings of the International Computer Music Conference (ICMC)*, 1996.

[2] W. Brent, "A timbre analysis and classification toolkit for pure data," in *Proceedings of the International Computer Music Conference (ICMC)*, 2009.

[3] N. Schnell, M. A. S. Cifuentes, and J.-P. Lambert, "First steps in relaxed real-time typo-morphological audio analysis/synthesis," in *Proceedings of an International Conference on Sound and Music Computing (SMC)*, 2010.

[4] D. Schwarz and N. Schnell, "A modular sound descriptor analysis framework for relaxed-real-time applications," in *Proceedings of an International Computer Music Conference (ICMC)*, 2010.

[5] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-time corpus-based concatenative synthesis with catart," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2006.

[6] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, and R. Borghesi, "Mubu & friends - assembling tools for content based real-time interactive audio processing in max/msp," in *Proceedings of an International Computer Music Conference (ICMC)*, 2009.

[7] M. Puckette, T. Apel, and D. Zicarelli, "Real-time audio analysis tools for pd and max," in *Proceedings of the International Computer Music Conference (ICMC)*, 1998.

[8] W. Brent, "Cepstral analysis tools for percussive timbre identification," in *Proceedings of International Pd Conference (PdCon09)*, 2009.