

# UN MÉTODO HEURÍSTICO PARA OBTENER LA ENVOLVENTE DE SEÑAL Y SU IMPLEMENTACIÓN DE SOFTWARE

## A HEURISTIC APPROACH TO OBTAIN SIGNAL ENVELOPE WITH A SIMPLE SOFTWARE IMPLEMENTATION

C. Jarne\*<sup>1,2</sup>

1 Universidad Nacional de Quilmes - Departamento de Ciencia y Tecnología

2 CONICET

\*cecilia.jarne@unq.edu.ar

Recibido: 18/04/18; aceptado 16/05/18

---

Signal amplitude envelope allows to obtain information of the signal features for different applications. It is widely used to pre-process sound and other signals of physiological origin in human or animal studies. In order to obtain signal envelope, a fast and simple algorithm is proposed based on peak detection.

The procedure presented here is quite straightforward and can be used in different applications of time series analysis. It can be applied in signals with different origin and frequency content. This algorithm is implemented based on python libraries. The open source code is also provided. Aspects on the parameter selection are discussed to adapt the same method for different applications. Also traditional methods are revisited and compared with the one proposed here.

*Keywords: Algorithm, Signal analysis, Envelope, Rich spectral content, Python code, Open source*

---

### Introduction

In audio signals, envelope is usually defined together with temporal fine structure. The amplitude modulations of animal sounds, speech or musical instruments are important features that often can be used to understand the physics or biology behind different processes. Accurate estimation of the amplitude, or equivalently energy, or envelope of a time-domain signal (waveform) is not trivial. Ideally, the amplitude envelope should outline the waveform connecting the main peaks and avoiding over tting as was pointed out in.<sup>1</sup>

The definition of envelope in the literature is often ambiguous and lacks of an exact mathematical definition. It is commonly agreed that envelope varies slowly and, in some empirical view, it should pass the prominent peaks of the data smoothly.<sup>2</sup>

Amplitude envelope is a relevant acoustic feature. It allows to get a reliable estimation that follows closely sudden variations in amplitude and avoids ripples in more stable regions. Depending on the fundamental frequency of the signal, it is possible to have a near optimal order selection in some regions. Amplitude variations depending on the application allows to obtain information on the signal patterns or features. i. e. peak detection, maximum, signal sudden variations, etc.

In addition to sound signals, the signal envelope calculation (or similar) is useful in many application of signal analysis in different fields such as biology, medicine, geology, material sciences, among others.<sup>3</sup> 10A simple algorithm together with a code implementation could became a useful tool to be provide for experimental analysis across different disciplines.

Currently to obtain signal envelope there is a diversity of standard methods implemented in different programming languages. Some of the most sophisticated has not been developed in open source code. On the other hand, most of the traditional methods do not allow to obtain a not attenuated envelope. Here a method to obtain the signal envelope is proposed and described. Additionally an open source implementation in python code is provided.

This work is organized as follows: Section II is used to provide a review of existing and most popular techniques. These techniques were also implemented with python libraries and compare with the algorithm proposed here. A comment on the more sophisticated algorithms is also presented. The implementation of these methods is not justified in relation to the programming effort versus the gain in processing. Also the fact of these works does not deal with standard methods, whose implementations are not available in open source code, is a point against the philosophy of present work. The algorithms are not straightforward to implement and the analysis is beyond the scope of present work. The main purpose of this works is to propose a simple algorithm to get the amplitude signal envelope in any situation with a direct implementation. In Section III the proposed algorithm is described. The implementation is explained in Section IV and some examples of application are provided in Section V. This section includes the discussion of how to chose the parameters in order to adapt the method for

different signals sources and some cues to lead to good practices in signal analysis are commented. The conclusion are presented in section VI. The code is available at the local University repository. In Section VI the reference to download the code is provided.

## **Current envelope techniques**

### **Classical techniques**

The classical approach for the estimation of the amplitude envelope of a time-domain signal is the technique known as envelope follower.<sup>11, 12</sup> This method has been widely used and has several implementations across different programming languages and also analog circuits. It consists basically of rectifying the waveform and then low-pass filtering it. This can be implemented in the analog or digital domain. With programming tools, the algorithm consist of taking the absolute value of the digitalized signal, and then use a digital low-pass filter. Several programming languages has libraries with build-in filters such as MATLAB<sup>13</sup> or Python.<sup>14</sup> For the comparison that is presented in further in Section III we used filters from python libraries.

Another interesting direct approach is to calculate the instantaneous root mean square (RMS) value of the waveform through a sliding window with nite support.<sup>15</sup> Some authors have proposed other techniques to obtain a more reliable estimation of the amplitude envelope of waveforms. An early attempt<sup>12</sup> consisted of a piece-wise linear approximation of the waveform. The amplitude envelope is created by nding and connecting the peaks of the waveform in a window that moves through the data. Python libraries were used to generate a function to smooth the signal via RMS together with the sliding window and then compare it with the implementation of the method corresponding to the present work (Section III).

Other classical technique consists of using the analytic signal from Hilbert Transform. Notably, if the Hilbert transform of  $S(t)$  is equal to its quadrature signal, then the estimates are equal to the actual information signals.<sup>11</sup> Synthetic signals (i.e. AM) can be constructed to have this property, but there is no reason to expect that animal sounds, acoustic musical instrument sounds or speech also present it. A more realistic condition is verified when we are dealing with narrow-band signals.<sup>16</sup> In sounds with rich spectral content it is not possible to use this approach because that condition is not fulfilled.

A straightforward approach is also an algorithm that directly use the prominent peaks and finally get the envelope by interpolation. A detail of this method can be found in.<sup>2</sup> It is an iterative process that needs to add noise to the original signal. In<sup>2</sup> not open source code is provide.

### **Regarding more sophisticated techniques**

A sophisticated method developed with not attenuated envelope is called Cepstral Smoothing.<sup>1</sup> It uses the real cepstrum defined as the inverse Fourier transform of the log magnitude spectrum. Regarding the log magnitude spectrum of a signal, it is possible to interpret each cepstral coefficient as a measure of the energy present in discrete frequency bands of that signal. Low-pass filtering the cepstrum would result in a smoother version of the log magnitude spectrum. If we only want to represent the spectral envelope, discarding information about the partials we should set the cutoff frequency below the period of the signal. This method is more sophisticated than the standard techniques and was discussed in detail in,<sup>1</sup> but not software implementation has been provided either.

Another method is called empirical mode decomposition (EMD)<sup>17</sup> allows to obtain the envelope as a result of an optimization process and sought as a minimum of a quadratic cost function. EMD is an adaptive data analysis method for analyzing nonlinear and non-stationary data, which was proposed by Huang et al. in 1998. A solution of this optimization problem is obtained in this study, and it is shown that the method could be widely implemented by choosing free parameters, where the frequency resolution or the number of intrinsic mode functions (IMFs) are tuned as well as the shape of the envelopes.

Both method showed to be interesting and accurate. Nevertheless as explain in Section I a disadvantage is that there are not published open source code implementations or libraries of these more sophisticated envelope techniques in any programming language of both works. This prevent us to perform the direct comparison of these algorithms with standard techniques or the one proposed in this paper without having to develop and debugging additional code.

In following section the algorithm developed for present work is described.

### **Current method: an envelope based on a direct peak detection**

To obtain an envelope output that will not be attenuated, a possible approach is to use a peak detection algorithm combined with a moving window. In the approach presented here, every value in the small window bunch is replaced by the maximum.

Later it is possible to lowpass-filter the resulting signal afterwards to get rid of the remaining staircase. This process is summarized in the schematic presented in Figure 1. In more detail, the method implemented in present work consist of three simple steps. The steps are very basic and allows to obtain a smooth signal following the amplitude variations as precisely as required.

-**The first step** is to take absolute value of signal  $S(t)$ , meaning  $|S(t)|$  as it is shown in Figure 2 top panel.

-**In the second step** it is necessary to divide the  $|S(t)|$  in  $k$  in non overlapping bunches of  $N$  samples:

$$|S(t)| = S_1(t) + \dots + S_i(t) + \dots + S_k(t), \quad (1)$$

with each  $S_i(t) > 0$ . Then the maximum amplitude value is taken from the signal in each bunch:  $Max(|S_i(t)|) = M_i$ , corresponding to the  $j$  element value in each bunch (peak detection). At this point each value in the bunch is replaced by the maximum  $M_i$ . In this step the signal  $R(t)$  looks like Fig. 2 middle panel.

-**The third step** consist of a lowpass-filter applied to the resulted signal  $R(t)$  to get rid of the remaining staircase ripple to obtain  $\hat{S}(t)$ , as illustrated in the Figure 2 bottom panel.

The signal obtained with this technique has the same sample rate that the original signal. This could be very useful in order to compare the envelope with other simultaneous measurements related to the original signal.

By using this three-step method the envelope obtained follows the signal amplitude variations without attenuation, if the parameters are carefully selected as shown in Fig.2. In this example the filter cutoff frequency is chosen between 100 - 150 Hz to ensure that the signal envelope follows signal variations as rapid as 10 mS combined with the window bunch size of 35 samples. These two variables could be adjusted to follow variations as rapid as needed, depending on the required signal application.

A comparison between the classical implementations of envelope methods (the envelope follower and RMS approach) and the one proposed in this work is presented in Figure 3. The code provided in section VI allows to compared these methods for any audio signal of any sample rate and frequency content. In the example selected for the figure, the frequency cut for the envelope follower approach in this example is 150 Hz. For the RMS mothod the slide window is 50 samples. For the three-step algorithm parameters are selected as in Figure 2. Main difference between standard methods and the one proposed here is regarding the attenuation.

## Discussion and Software implementation

The previous section was a general description of the method presented. Some particular considerations that has to be taken into account for the code implementation are discussed in this section. In particular the libraries that ware used, the functions and the selection of the parameters in the code, there are also presented as the arguments for the choices that ware made.

The language used for the implementation was Python.<sup>14</sup> The main reason to use python is the expansive library of open source data analysis tools, also the web frameworks and the testing instruments that make one of the largest current programming community. Another reason is that Python is an accessible language for new programmers and the community provides many introductory resources.

Regarding the implementation parameters, the size of the bunch for the signal to be divided is an important one. It will mainly depend on the sampling rate  $f_s$  but also on the frequency content and time scale variations that the user wants to capture from the original signal. For the examples presented here sample rate is 44:100 KHz and frequency content is between 100 Hz and 10000 Hz. In this case a proper bunch size is between 20-200 samples. To chose larger bunch size means that signal envelope means that the envelope deviates from the amplitude of the original signal but very small bunches means that user will capture small ripple variations of the signal.

Since we have audio les available to analyze, the Scipy Signal processing library functions was used to get the audio sample rate and signal vector and write a code with these input variables.

With respect to the digital filter, it is necessary to choose a filter that does not produce phase shift differences (a zero-phase filtering. It means that does not shift the signal as it filters. Since the phase is zero at all frequencies, it is also linear-phase.

Filtering backwards in time requires to predict the future signal value. This kind of filters can not be used for "online" real-life applications, only for off-line processing of recordings of signals or if we filter signal with some small-delay as it is acquired.

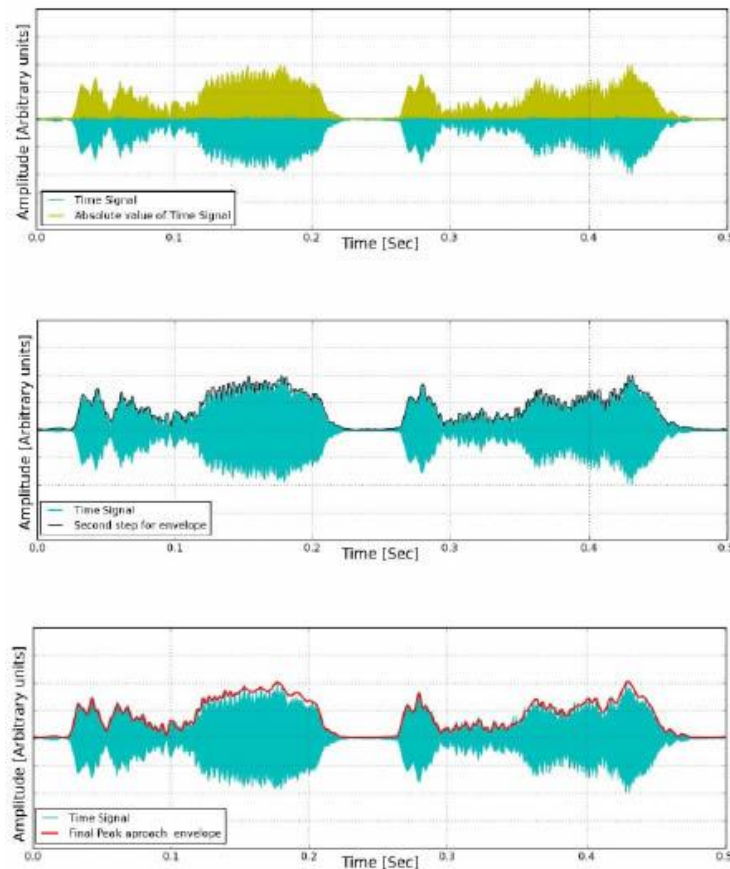
The scipy signal library<sup>18</sup> has two kinds of functions to choose for filter design. One is called `scipy.signal.lfilter` and the other `scipy.signal.lfilt`. The first one (`lfilter`) is causal forward-in-time filtering only, similar to a real-life electronic filter. It cannot be zero-phase. It usually adds different amounts of delay at different frequencies. The appropriated method for the off-line application is to use `filtfilt` with no shift in the signal as it filters.

Once the zero-phase filter is selected, the following step consists of choosing the type of filter as well as the frequency behavior. In this case we used a Butterworth low pass filter type `scipy.signal.butter` of fourth order. Frequency attenuation (behavior) of this filter is elsewhere.<sup>19</sup> The other key point to obtain the envelope appropriate for the specific application is to choose the frequency cut of the low pass filter from Third step in the algorithm. Depending on this value you can obtain as many detail on the amplitude as it is required.

The method proposed here is not computationally expensive, so it can be applied in different systems.



**Figure 1:** Algorithm schema. Input signal is called  $S(t)$  and the envelope obtained is  $\hat{S}(t)$ . Each arrow represents one step of the algorithm.



**Figure 2:** Zoom over a song canary segment. Top: Absolute value of time signal. Middle: Peak detection algorithm (pre-envelope).

Bottom: envelope, the upper signal when a low pass filter is applied. Filter cutoff frequency is 120 KHz and window bunch size of 35 samples.

For instance for 1.5 second of signal recording with a sample rate of 44:1 KHz, the computational time needed to run the code is < 0:5 Sec.

To summarize, the algorithm keys for the implementation are: proper bunch size, chose a zero-phase filter, and chose a frequency cut to follow the signal changes as required.

### **Results: example of the implementation over different sounds and parameter selection**

Four different sounds were selected from public data bases to test the algorithm. The envelope result from the method implemented is shown over different kinds of sound segments with or without rapid variations. Audio files was selected from: two sounds of very different animals presented in Figure 4 (canary song and a whale song), human speech (English language) and an instrument recording of piano song presented with corresponding spectrogram. Table 1 summarizes the parameter values for each case. Sound files and code are provided as supplementary material.

The examples was selected with different time scales as well as frequency content. The parameter selection will depend on the user needs. A heuristic search for the values convenient in a particular application will depend on how many detail is required to follow the signal variation.

A good method must be flexible enough to be applied to different kinds of signals. In this case the focus was on sound signal, but it is not exclusive. The parameters of the algorithm can be selected depending on the application and the type of signal to be applied.

### **Conclusions**

In this work a heuristic and simple technique is proposed for envelope estimation and a open source code is provided. This algorithm used is very simple to implement and it is not just another empirical method. Even when the procedure is straightforward and simple, its robust solution for different signals and required resolution on signal detail. An important advantage of this method is that it can be applied insignal with different spectral content. The parameters can be adjusted for the time scaling required by the user in order to select the smoothness of the estimated envelope. Experimental applications presented shows that the algorithm can achieve good quality on the temporal envelope estimation of data. It was observed that the estimated envelope can track quite well the changing trend of input data in different time scales.

This algorithm can be implemented in off-line processing applications, e.g. animal sounds (particularly bird songs), musical instrument sound analysis, heart sound analysis, vibration signal analysis, and etc. In this paper, temporal audio signals were used as examples. However, there are no requirements on the specific domain of the data. It is appropriate for one dimensional data (e.i. time domain, frequency domain, space domain, etc.). Additionally a python implementation is presented to be used with any sound input.

A further key to apply the algorithm in almost real-time implementation, is to process signal time series in segments larger than 1 bunch size. It could be useful depending on the sample rate of the digitalized signal.

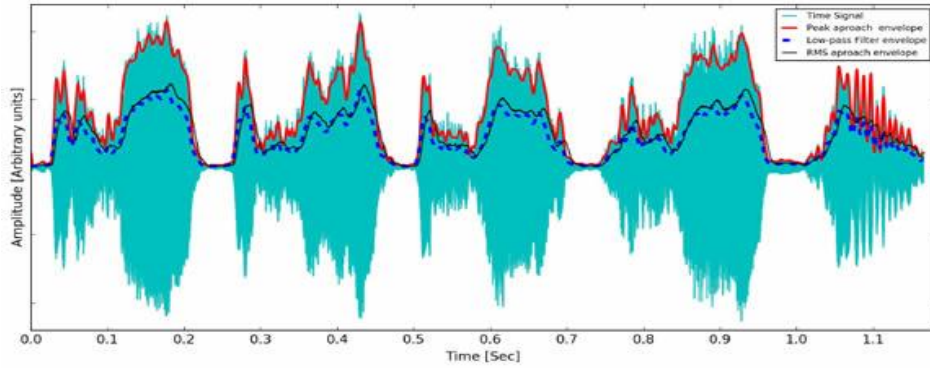
### **Availability of data and source code**

The software was developed on python. Code is available under password envelope at University local repository:

<http://ceciliajarne.web.unq.edu.ar/investigacion>

and

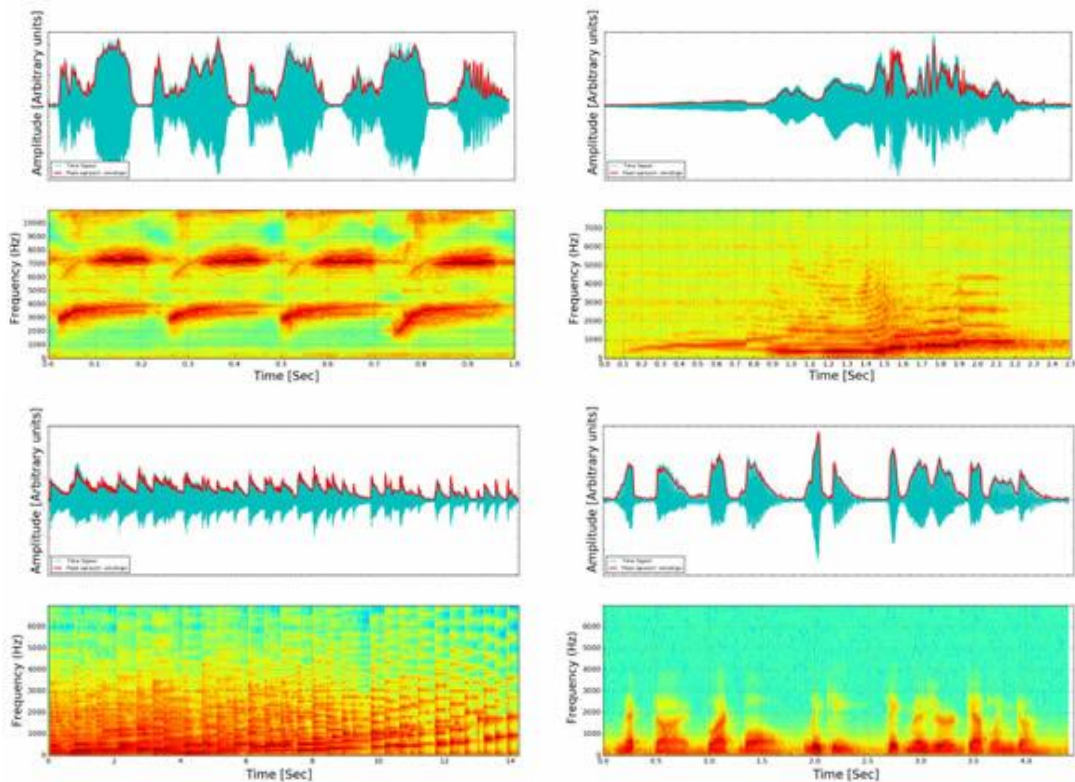
<https://github.com/katejarne/Envelope Code>



**Figure 3:** Comparison between the classical implementations of envelope methods and the one proposed in this work. Blue dashed line for low-pass filtered envelope, black thin line corresponds to RMS slide window and red line the proposed method.

Source	bunch size[samples]	Cut Frequency [Hz]
Canary sound <sup>21</sup>	35	300
Whale sound <sup>22</sup>	50	300
Words spoken in English <sup>23</sup>	50	100
Piano stair scale notes <sup>24</sup>	200	100

**Table 1:** Examples for the values of the parameters selected for samples of different kind of sound origin. Sample rate is 44:1 KHz for all segments in the table.



**Figure 4:** Up Left: Piano song segment where the frequency of the played notes increases over time and the corresponding sonogram. Up Right: Segment of spoken words in English. Down Left: A canary song signal with the estimated envelope and its sonogram. Down Right: A whale song with the corresponding envelope. Frequency content of both signals and also time scales are very different, although the method works in both cases.

## References



1. M. Caetano, and X. Rodet. Improved estimation of the amplitude envelope of time-domain signals using true envelope cepstral smoothing. 2011 IEEE Int. Con. on Acoustics, Speech and Signal Processing (ICASSP) page 4244-4247. DOI : [10:1109=ICASSP:2011:5947290](https://doi.org/10.1109/ICASSP.2011.5947290).
2. Q. Meng, M. Yuan, Z. Yang. An empirical envelope estimation algorithm. 2013 6th International Congress on Image and Signal Processing (CISP). doi : [10:1109/CISP:2013:6745226](https://doi.org/10.1109/CISP.2013.6745226).
3. Michael J. Crosse, Edmund C. Lalor. The cortical representation of the speech envelope is earlier for audiovisual speech than audio speech. J Neurophysiol 111: 1400-1408, 2014; doi : [10:1152/jn:00690:2013](https://doi.org/10.1152/jn.00690.2013).
4. S. Tilsen, A. Arvaniti. Speech rhythm analysis with decomposition of the amplitude envelope: characterizing rhythmic patterns within and across languages. The Journal of the Acoustical Society of America 134 (1), 628-639. DOI : [http://dx.doi.org/10:1121/1:4807565](https://doi.org/10.1121/1.4807565).
5. M. Feldman. Hilbert transform in vibration analysis. Volume 25, Issue 3, April 2011, Pages 735-802. [http://dx.doi.org/10:1016/j:ymssp:2010:07:018](https://doi.org/10.1016/j.ymssp.2010.07.018).
6. A. Veltcheva, P. Cavaco, C. Guedes Soares. Comparison of methods for calculation of the wave envelope Ocean Engineering Volume 30, Issue 7, May 2003, Pages 937-948. [http://dx.doi.org/10:1016=S0029-8018\(02\)00069-0](https://doi.org/10.1016/S0029-8018(02)00069-0).
7. J. Liu, W. Liu, H. Wang, T. Tao, J. Zhang. A novel envelope extraction method for multichannel heart sounds signal detection. International Proc. of Computer Science & Information Tech; 2012, Vol. 51, p630 DOI : [10:7763/IPCSIT:2012:V51:106](https://doi.org/10.7763/IPCSIT:2012:V51:106).
8. Wu-bin Li, Chang-hou Lu, Jian-chuan Zhang. A lower envelope Weber contrast detection algorithm for steelbar surface pit defects Optics & Laser Technology 45 (2013) 654-659. [http://dx.doi.org/10:1016/j:optlastec:2012:05:016](https://doi.org/10.1016/j.optlastec.2012.05.016).
9. S. Choi, Z. Jiang. Comparison of envelope extraction algorithms for cardiac sound signal segmentation. Expert Systems with Applications 34 (2008) 1056-1069. [doi:10:1016/j:eswa:2006.12:015](https://doi.org/10.1016/j.eswa.2006.12.015).
10. Wilsch A., Neuling T., Obleser J., Herrmann C.S. Transcranial alternating current stimulation with speech envelopes modulates speech comprehension. Neuroimage 2018. DOI : [10:1016/j:neuroimage:2018:01:038](https://doi.org/10.1016/j.neuroimage.2018.01.038)
11. A. Potamianos, P. Maragos. A comparison of the energy operator and the Hilbert transform approach to signal and speech demodulation Signal Processing 37 (1994) 95-120.
12. A. Schloss. On the Automatic Transcription of Percussive Music - From Acoustic Signal to High- Level Analysis, Stanford University, Ph.D. Thesis 1985.
13. MATLAB 6.1, The MathWorks Inc., Natick, MA, 2000
14. G. van Rossum. Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI). <https://www.python.org/>. Amsterdam, May 1995.
15. J. Hajda. A New Model for Segmenting the Envelope of Musical Signals: The relative Saliency of Steady State versus Attack, Revisited, J. AES, Nov. 1996.
16. W.M. Hartmann. Signals, Sound, and Sensation . Springer-Verlag-AIP Press, 1997.
17. Y. Washizawa, T. Tanaka, Danilo, P. M. Cichocki. A Flexible Method for Envelope Estimation in Empirical Mode Decomposition. Int. Con. on Knowledge-Based and Intelligent Information and Engineering Systems KES 2006. pp 1248-1255. DOI : 10:1007/11893011158.
18. Travis Oliphant, Pearu Peterson, Eric Jones 2001 Scipy Library <http://scipy.org/>
19. Butterworth digital and analog filter design library. scipy - 0:14/reference/generated/
20. Python Signal Processing Libraries. [https://docs.scipy.org/doc/scipy - 0:14:0/reference/](https://docs.scipy.org/doc/scipy-0:14:0/reference/)
21. Pianosound from: UCLA Phonetics Lab Language Archive [http://archive.phonetics:ucla:edu/Language/](http://archive.phonetics.ucla.edu/Language/)
22. Canary sound from canary examples for training: <https://www.youtube.com/watch?v=qnZ-gj7Si2w&t=21s>
23. Whale sound from The Oceania Project Byron Bay, Australia. <https://soundcloud.com/iwhales>
24. Piano sound from Alexander Lerch <http://www.audiocontentanalysis.org/>