PREDOMINANT PITCH EXTRACTION IN POLYPHONIC SIGNALS
PERFORMANCE AND ACCURACY EVALUATION OF ESSENTIA AND THE ISMIR2004 CONTEST WINNER

Marc Sanchez Martinez
Universitat Pompeu Fabra
Music Technology Group
marc.sanchez10@estudiant.upf.edu

Victor Badenas Crespo
Universitat Pompeu Fabra
Music Technology Group
victor.badenas01@estudiant.upf.edu

ABSTRACT

Music in general is **polyphonic** (multiple pitches sounding at the same time) and there usually is a melodic line that stands above the rest of the sounds. In this paper, a study of **Rui Pedro Paiva's** proposed structure for Predominant Pitch Extraction is compared against Essentia's **PredominantPitchMelodia** and an implementation of the authors based on Paiva's structural algorithm.
These three algorithms have the same goal, which is to detect and isolate the main melodic line from the rest. In the end we conclude that the essentia algorithm is, by today's standards the best in terms of performance and accuracy and we propose some possible lines of further investigation for the work done in this paper.

1. INTRODUCTION

The field of **polyphonic pitch extraction** has been an important topic in Music Information Retrieval with a big influence in the researcher's community in the 2000's and the early 2010's. Numerous algorithms [4] [3] have been written and developed for the purpose of detecting the pitch of the main voice in a recording where there is background noise (tonal or stochastic). Those algorithms used **pitch detection algorithms** (PDA), which are algorithms designed to estimate the pitch or fundamental frequency of an oscillating signal. This can be done in the time domain, the frequency domain, or both.

In the **ISMIR 2004** conference, a first try to do a **contest** [1] on predominant pitch extraction was done by the MTG-UPF, specially by Gomez, Emilia and Ong, Beesuan and Streich, Sebastian. The task of the contest was to analyze a dataset provided by the MTG and get the highest accuracy possible compared to a reference. Four participants entered the contest of which **Rui Pedro Paiva's** entry won. The time of execution of each of the algorithms was computed as well.

The main objective of this work is to evaluate and try to reproduce Rui Pedro Paiva's entry to compare it to the MTG's own PredominantPitchMelodia in the Essentia Library, which is based in Salamon, J., & Gomez, E.'s paper [4].

2. STATE OF THE ART

PDAs typically estimate the period of a quasiperiodic signal, then inverts that value to give the frequency.

The simplest approach to measure pitch predominance is to use zero crossing points or more efficiently, the **zero crossing rate** Eqn (1) which is the frequency of the sign shifts in a signal's time domain. However, this does not work well with complicated waveforms which are composed of multiple sine waves with differing periods or noisy data.

$$Z_n = \sum_{m=-\infty}^{\infty} \mid sgn[x(m)] - sgn[x(m-1)] \mid w(n-m) \tag{1}$$

Other more sophisticated approaches compare segments of the signal with other segments offset by a trial period to find a match. **Autocorrelation** Eqn (2) algorithms work this way. These algorithms can give quite accurate results for **highly periodic signals**. However, they have false detection problems like "octave errors" and can sometimes cope badly with noisy signals and do not deal well with polyphonic melodies.

$$r_x(\tau) = R_{XX}(\tau) = E[X_t \overline{X_{t+\tau}}] = \sum_{n \in \mathbb{Z}} x(n)\overline{x(n-\tau)} \tag{2}$$

3. METHODOLOGY

As Paiva's paper from 2005 is not highly reproducible due to the lack of a repository or a detailed explanation of the algorithm used in their proposal, the implementation proposed in the Jupyter Notebook [1] is based on Figure 1 which is the flow diagram that was proposed in Paiva's paper but differs in the core implementation of each block.

3.1 Multi Pitch Detection (MPD)

The MPD is done frame by frame: in each iteration the autocorrelation is computed and normalized, all possible

[1] https://github.com/MarcSM/predominant-pitch-extraction-in-polyphonic-signals

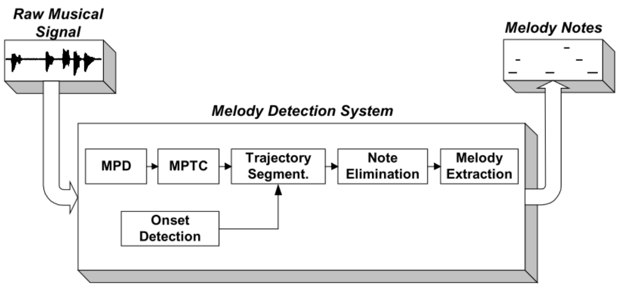

Figure 1. Paiva's flow diagram taken from [2]

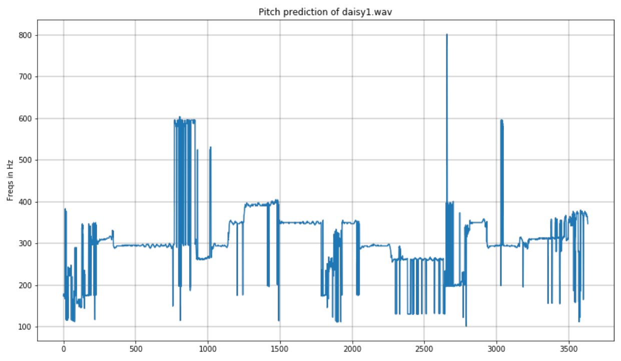

Figure 2. Output of the MPD algorithm

maxima of the autocorrelation are considered. Then the peaks of the signal are found and they are appended to a list of frequencies which is sorted by the probability of them of being the predominant pitch in that time frame. The output is Figure 2.

The method chosen to do this was autocorrelation because, even though autocorrelation is a function that does not work well with polyphonic melodies, it is proven to be a good tool when extraction of monophonic pitches are the desired result.

3.2 Multi-Pitch Trajectory Construction (MPTC)

The MPTC gets the pitch with the best autocorrelation in each frame after replacing with zeros the frames where no important information about pitch is found. Then it performs an octave correction which basically corrects errors of an octave shift from frame to frame. The output of the MPTC algorithm is Figure 3

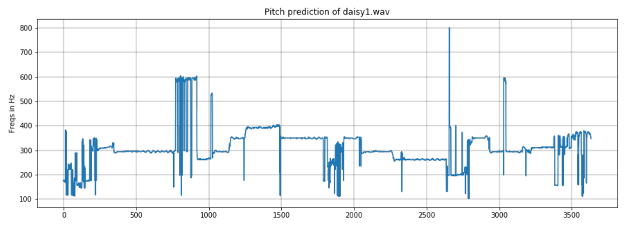

Figure 3. Output of the MPTC algorithm

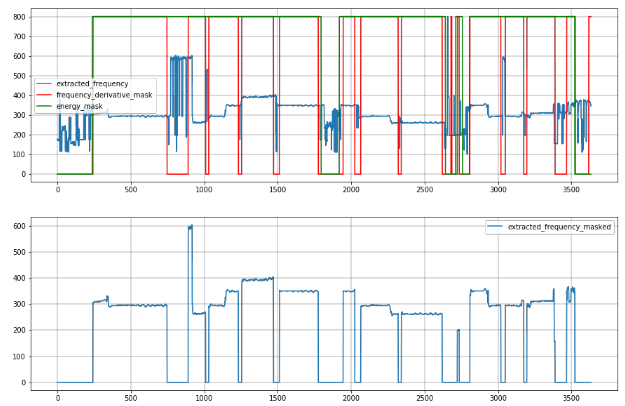

Figure 4. Output of the Trajectory Segmentation algorithm

3.3 Trajectory Segmentation

It creates a set of different masks that decide whether the main pitch line is playing or not. The masks that are being applied are:

- Energy envelope per frame: computes the normalised energy value per frame of the audio signal, and, after some smoothing, it applies a threshold to know where to apply the mask.
- Frequency (un-quantised midi values for linearity of the system) derivative per frame: computes the derivative of the log(frequency) or midi values, smoothens it and applies a threshold to know where to apply the mask.

After computing the masks they are applied to the extracted pitch. Figure 4

3.4 Evaluation Method

In the ISMIR2004 conference, two evaluation functions coded in MATLAB were used to evaluate the performance of the algorithms. Those functions were translated to python and then used to evaluate the performance of the algorithms.

4. RESULTS

Essentia's PredominantPitchMelodia algorithm took 0.39 s per sound file in average, the algorithm proposed in this paper based on Paiva's algorithm took 16.6 s per sound file in average and Paiva's 2004 algorithm took 3346,67s in average to compute.

Obviously, the comparison is not really relevant as the essentia algorithm and the algorithm of this paper were executed in single-thread mode in an Intel Core i5-8400 @ 2.8GHz, Mac OS environment, and are coded using C++ and Python languages respectively and Paiva's algorithm was run in an Intel Pentium @ 1.2 GHz, Windows environment, and was coded in Matlab (Java).

Once the algorithms were run, a comparison is performed where the accuracy achieved by Paiva's algorithm

	evalOption1	evalOption2	**Average**
daisy1	26.32	26.45	**26.38**
daisy2	15.18	15.21	**15.20**
daisy3	-11.50	-11.47	**-11.48**
daisy4	-3.09	-3.09	**-3.09**
jazz1	18.29	20.11	**19.20**
jazz2	5.77	5.22	**5.47**
jazz3	35.80	35.80	**35.80**
jazz4	-4.29	6.23	**0.97**
midi1	-40.61	-37.91	**-39.25**
midi2	13.18	13.38	**13.28**
midi3	6.28	9.75	**8.01**
midi4	-15.91	-11.65	**-13.78**
opera_fem2	42.97	46.15	**44.56**
opera_fem4	33.78	34.56	**34.17**
opera_male3	34.79	37.35	**36.07**
opera_male5	12.94	35.17	**24.05**
pop1	-5.56	-5.57	**-5.56**
pop2	7.17	7.10	**7.13**
pop3	-16.55	-3.04	**-9.79**
pop4	8.97	12.16	**10.56**
Average	**8.19**	**11.60**	**9.89**

Table 1. Essentia Winner comparison Table

is substracted from the accuracy of every sound file and algorithm. The resulting data are Table 1 and Table 2. A positive value represents that the essentia or the extracted algorithm is better. A negative value represents otherwise. Also the difference of the essentia and the proposed algorithm from this paper is computed as seen in Table 3.

5. CONCLUSIONS

Despite being a paper of 2004, the fact that the algorithm it is not public has forced us to do some research about pitch extraction and to implement our own code based on the paper's diagram, learning within the proces. Our results are worse than the Essentia algorithm, but is actually better than the winner of the 2004 competition, thus this algorithm can be taken as a baseline to try to build a better algorithm and try to get a higher accuracy.

6. FUTURE WORK

We know that this paper is from 2004 so probably some of the things that we are going to propose has already been done plus some of them are things that we were not capable to reproduce due to the low reproducibility of the paper. So, the first thing to do would be to complete the algorithm with the missing parts of the main diagram Figure 1, which are the onset detection, note elimination and the melody extraction. It will be good to implement the melody extraction because right now the algorithm does not have any heuristic on how to extract the melody as we are getting the raw extracted pitch as the final result. Finally another thing that is missing is to perform some kind of harmonic correction in the MPTC module.

	evalOption1	evalOption2	Average
daisy1	21.78	22.89	**22.33**
daisy2	11.57	11.72	**11.65**
daisy3	-20.88	-8.48	**-14.68**
daisy4	-15.78	-7.77	**-11.78**
jazz1	8.90	11.02	**9.96**
jazz2	-2.34	1.49	**-0.46**
jazz3	25.26	29.89	**27.58**
jazz4	-1.00	6.59	**2.80**
midi1	-38.02	-26.10	**-32.05**
midi2	1.11	1.84	**1.48**
midi3	-4.88	-3.07	**-3.98**
midi4	-15.48	-4.90	**-10.18**
opera_fem2	28.85	32.85	**30.85**
opera_fem4	26.86	28.85	**27.85**
opera_male3	15.61	22.99	**19.30**
opera_male5	-0.76	13.36	**6.30**
pop1	-4.62	-3.40	**-4.01**
pop2	-8.46	-3.16	**-5.81**
pop3	0.10	2.30	**1.21**
pop4	8.22	10.97	**9.59**
Average	**1.80**	**6.99**	**4.40**

Table 2. Extracted Winner comparison Table

	evalOption1	evalOption2	**Average**
daisy1	4.54	3.56	**4.05**
daisy2	3.61	3.49	**3.55**
daisy3	9.38	-2.99	**3.20**
daisy4	12.70	4.68	**8.69**
jazz1	9.39	9.08	**9.24**
jazz2	8.11	3.74	**5.92**
jazz3	10.54	5.90	**8.22**
jazz4	-3.30	-0.36	**-1.83**
midi1	-2.58	-11.82	**-7.20**
midi2	12.07	11.54	**11.81**
midi3	11.16	12.82	**11.99**
midi4	-0.43	-6.75	**-3.59**
opera_fem2	14.12	13.30	**13.71**
opera_fem4	6.92	5.71	**6.32**
opera_male3	19.18	14.36	**16.77**
opera_male5	13.70	21.81	**17.75**
pop1	-0.93	-2.17	**-1.55**
pop2	15.63	10.26	**12.94**
pop3	-16.65	-5.34	**-11.00**
pop4	0.75	1.19	**0.97**
Average	**6.39**	**4.60**	**5.50**

Table 3. Essentia Extracted comparison Table

7. REFERENCES

[1] Ismir 2004 melody extraction contest. http://ismir2004.ismir.net/melody_contest/results.html. Accessed: 2019-02-27.

[2] Rui Pedro Paiva. On the detection of melody notes in polyphonic audio. 2005.

[3] Rui Pedro Paiva, Teresa Mendes, and Amílcar Cardoso. Melody Detection in Polyphonic Musical Signals: Exploiting Perceptual Rules, Note Salience, and Melodic Smoothness. *Computer Music Journal*, 30(4):80–98, 2006.

[4] Justin Salamon and Emilia Gomez. Melody extraction from polyphonic music signals using pitch contour characteristics. *IEEE Transactions on Audio, Speech and Language Processing*, 20(6):1759–1770, 2012.