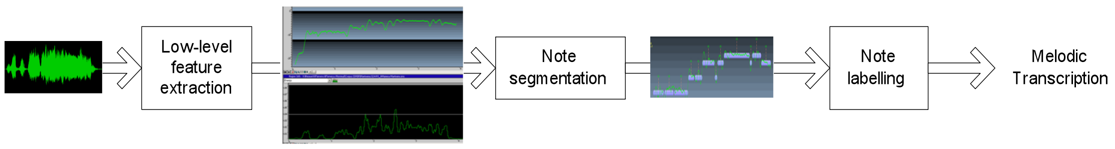
Lab 2: Melodic transcription

1. Introduction

The goal of this lab is to implement a simple melodic transcription method for drum loops or monophonic phrases by measuring variations of fundamental frequency, energy and spectral content, as shown in the next block diagram:



2. Material

- Software: Matlab/Octave.

- Fundamental frequency estimation method: TWM and yin algorithms (matlab/octave code from the Audio and Music Processing course), yin code by de Cheveigné and Kawahara (2002) (Please check *private.zip* for mac users), yin implementation available as vamp plugin (aubio http://www.vamp-plugins.org/download.html) or MELODIA software if you want to work with polyphonic music signals (Salamon and Gomez 2012) available at http://mtg.upf.edu/technologies/melodia. To use vamp plugins, you should output the information in txt format using Sonic Annotator (http://www.omras2.org/sonicannotator).

- Energy and spectral descriptors: you can take advantage of the yin ouput (pwr) and the code generated in Lab1 (e.g. spectral centroid, spectral flux) (ask your colleagues if you didn't work on that lab).

- Matlab utilities:

  - *vline.m*: `Draws a vertical line on the current axes at the location specified by 'x'.`
  - *onset.m*: simple segmentation algorithm based on variations of energy and fundamental frequency.

- Monophonic phrases from different instruments: 35 melodies from different instruments and 10 from different singing styles, in wav format. You can also enhance this set with monophonic melodies from freesound (e.g. drum loops).

- Polyphonic melodies of your own if you choose to work with MELODIA.

3. Description

3. A. Instantaneous fundamental frequency estimation.

Review the selected f0 estimation algorithm by reading the corresponding paper.

Visualize the result of f0 estimation for the proposed phrases. Comment on the accuracy of the method and the estimation errors that you find (for which instruments, ranges, missing notes, octave errors, etc). You can also try to tune the algorithm parameters, at least the fundamental frequency range or the window size[1].

Select 10 phrases where the fundamental frequency algorithm performs well enough to be used for note segmentation. Some examples of the estimated fundamental frequency are found in the Appendix. In case you cannot run any of the algorithm, I provide a txt file for each melody (*melody.wav.txt)*, containing yin output in 4 columns with the following values:

[time f0 ap pwr]

where

```
%   time: time in seconds
%   f0: fundamental frequency in octaves re: 440 Hz
%   ap: aperiodicity measure (ratio of aperiodic to total power)
%   pwr: period-smoothed instantaneous power (energy)
```

3. B. Note segmentation based on energy

Propose and implement an onset detection method based on detecting energy variations. You can follow the next steps:

1. Compute energy envelope. The yin algorithm outputs it, so you can read it from the 4th column of the txt file (*pwr*).
2. Take the log. Remember to add a small amount before in order to avoid a –Inf value.
3. Compute the energy derivate. You can use the Matlab function `diff` to compute the difference between consecutive values. You can also

[1] If you chose the win algorith, note that the output of the yin algorithm is measured in octaves (with respect to 440 Hz). Have a look at the file *yin.m* for more information regarding the used hop size and units.

compute the difference between the average of frame values before a given frame and the average of frame values after a given frame.

4. Locate the peaks of the log energy difference. Define a threshold parameter for selecting the peaks. You can re-use the peak estimation algorithm from SMS.

Display the audio signal, its energy, the log energy and the log energy difference. Try to perform 3 before 2.

Display also a vertical line for the selected peaks using the function *vline.m* for different threshold values. Evaluate the performance of this simple algorithm on the selected phrases (melodies or drum loops).

Think or implement some improvements to this basic method. For instance, consider using spectral descriptors from Lab1 to segment the phrases (try, for instance, with spectral flux or centroid). If you are dealing with drum loops, study the performance of the derivate of the energy on different bands to detect onsets corresponding to different percussive instruments.

3. C. Note segmentation based on fundamental frequency

Implement a basic note segmentation algorithm based on detecting variations in fundamental frequency. You can start from the code provided in the matlab file `onset.m`. You can perform some of the following steps (the order can vary and some steps might not be needed):

1. Conversion of fundamental frequency values in Hz to cents (you can assume that all phrases are tuned to 440 Hz).

2. Compute the fundamental frequency derivate. You can use the Matlab function `diff` to compute the difference between consecutive values. You can also compute the difference between the average of frame values before a given frame and the average of frame values after a given frame.

3. Peak picking on the decision function. Define a threshold parameter for selecting the peaks. You can re-use the peak estimation algorithm from SMS.

4. Consolidation/deletion of too short notes.

Other possible steps include:

5. Quantization of fundamental frequency values or derivate to 1 semitone resolution before computing derivate.

6. Grouping of frames with close fundamental frequency/pitch value.

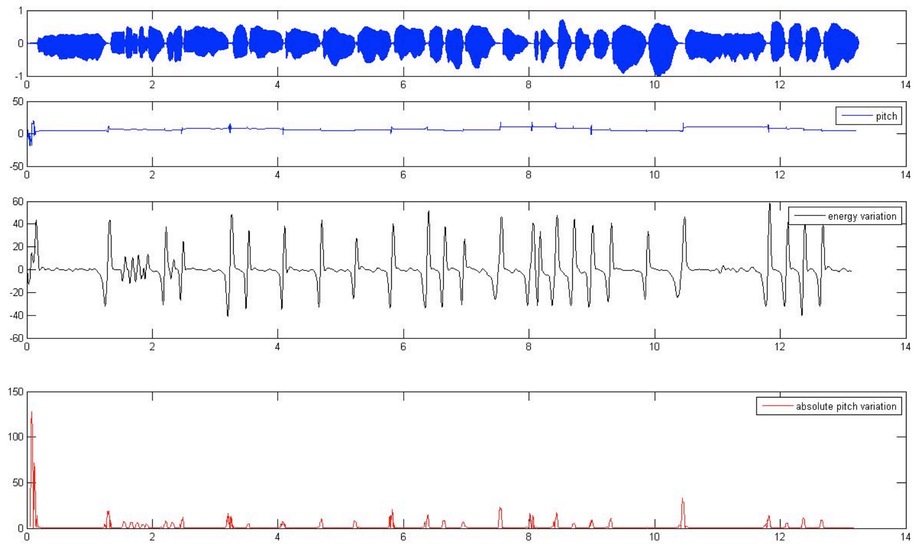
Display the audio signal, its fundamental frequency, and the smoothed fundamental frequency difference. Display also a vertical line for the selected peaks using the function *vline.m*.

Evaluate the performance of this algorithm on the selected phrases. Comment on the accuracy of the results and the limitations of the method.

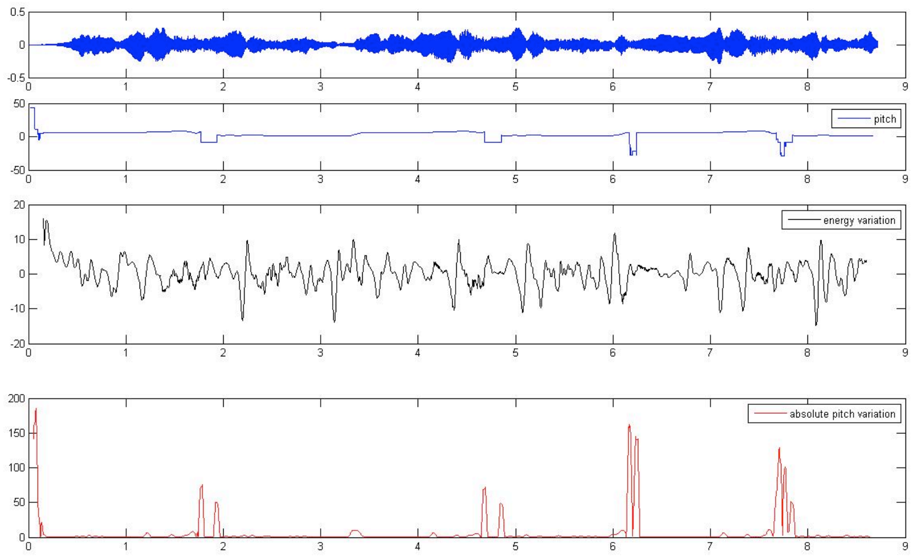
Think or implement some improvements to this basic method. For instance, consider the yin *aperiodicity* measure as a way to refine the fundamental frequency estimate, or use it for the segmentation to detect transitions between notes.

Examples of decision functions based on fundamental frequency and energy

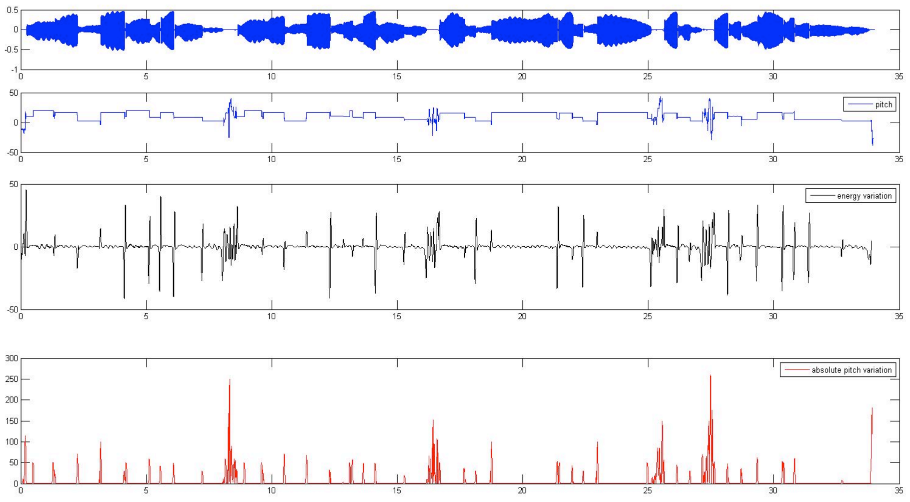
<u>Melody15</u>



<u>Melody16</u>



Melody25



3. D. Note segmentation based on both f0 and energy

Implement a function that merges the result from 3.A and C into a note segmentation algorithm. You can select the transitions detected by fundamental frequency and energy and then merge too short notes.

Display the audio signal, and a vertical line for the selected energy and fundamental frequency peaks using the function *vline.m*.

Evaluate the performance with the selected phrases and make comments regarding the limitations.

Think on possible ideas to merge the two decision functions into final note estimations.

3. F. Note descriptors

Compute the following descriptors for each detected note:

- Duration
- Fundamental frequency (median of f0 values over the note duration)
- Pitch (quantized fundamental frequency)

3. G. Global descriptors

Compute the following descriptors for each melody:

- Duration
- Number of notes
- Note density (number of notes per second)
- Pitch range

Implement at least one search example, e.g. "find the melody with highest note density".

4. Note segmentation evaluation strategy

In order to evaluate in an effective way the accuracy of the note segmentation algorithm, we can use the strategy proposed by the MIREX community. They propose to use the following evaluation measures:

- Note precision: ratio of correctly transcribed ground truth notes to the number of ground truth notes.
- Note recall: ratio of correctly transcribed ground truth notes to the number of transcribed notes.
- Note F-measure: harmonic mean of precision and recall (2*precision*recall/(precision+recall)).

We will consider the average value for the considered melodies. Here, a transcribed note is considered correct if its onset is within 50 ms of a ground-truth note and its f0 is within +- 50 cents of the corresponding ground-truth note. In addition, a correct computed note is required to have an offset value within 20% of the ground truth note offset, or within 50 ms whichever is larger. One ground truth note can only be associated with one transcribed note.

The file *mirex_note_eval.m* can help you to compute the mentioned measures. We should create between us the ground truth transcription for the melodies under study.

4. Schedule

- 1st week: implementation of note segmentation algorithm based on energy and f0
- 2nd week: note segmentation algorithm based on energy and f0, evaluation
- 3rd week: note descriptor and global descriptor implementation

5. Lab evaluation criteria

This lab is evaluated/10 points

- 6 points: correct implementation and study of the problem of note segmentation, limitations and challenges.
- 1 point: implementation of note descriptors
- 1 point: implementation of global descriptors
- 2 points: quality of the written report
- 1 point: best classification results or quality of work with respect to the group

References

- de Cheveigné, A., and Kawahara, H. (2002). "YIN, a fundamental frequency estimator for speech and music," J. Acoust. Soc. Am., accepted for publication (pdf)
- R. J. McNab, L. A. Smith, and I. H. Witten. *Signal processing for melody transcription*. SIG, Working paper, 95(22), 1996 http://www.cs.waikato.ac.nz/~ihw/papers/96RJM-LAS-IHW-Sig-Proc.pdf
- Dannenberg, R. B. and Hu, N. *Understanding search performance in Query-by-humming systems*, Proceedings of ISMIR, 2004. http://ismir2004.ismir.net/proceedings/p043-page-232-paper236.pdf

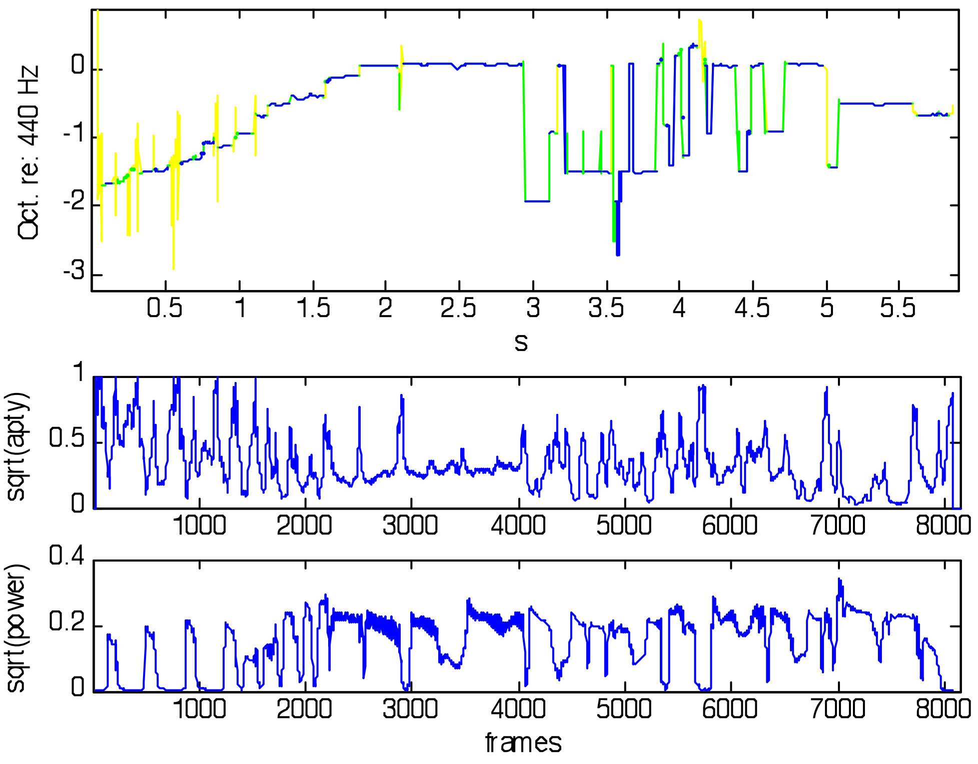
- J. Salamon and E. Gómez, "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics", IEEE Transactions on Audio, Speech and Language Processing, 20(6):1759-1770, Aug. 2012.

Appendix

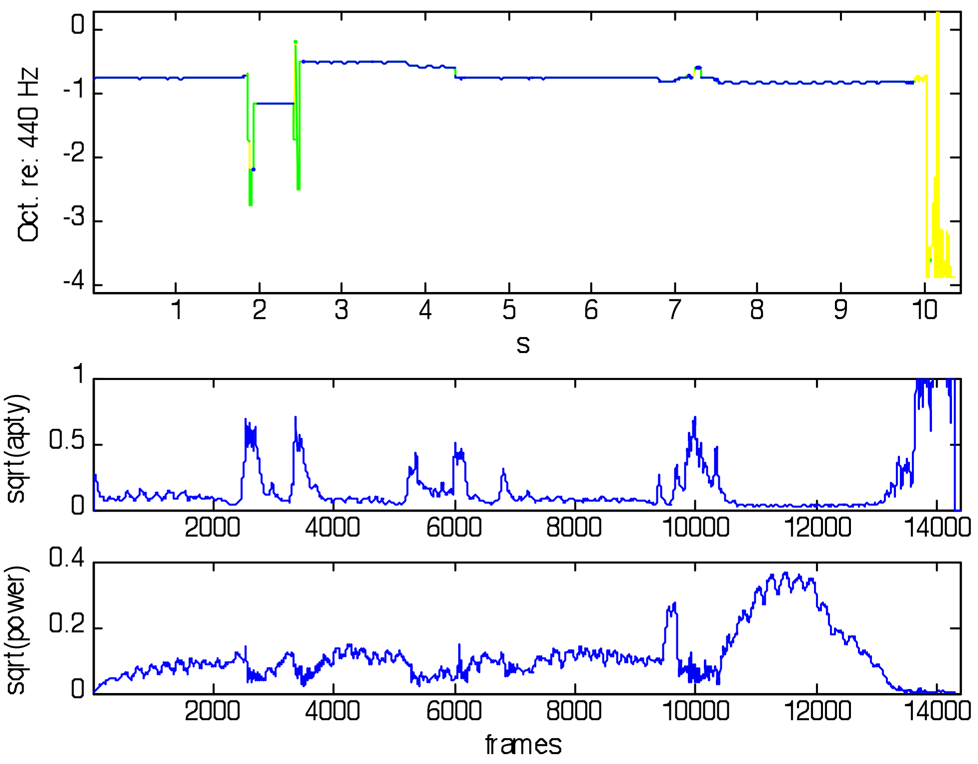
1. Fundamental frequency estimation using yin.

Fundamental frequency estimation using yin for some of the proposed melodies and using default parameters:

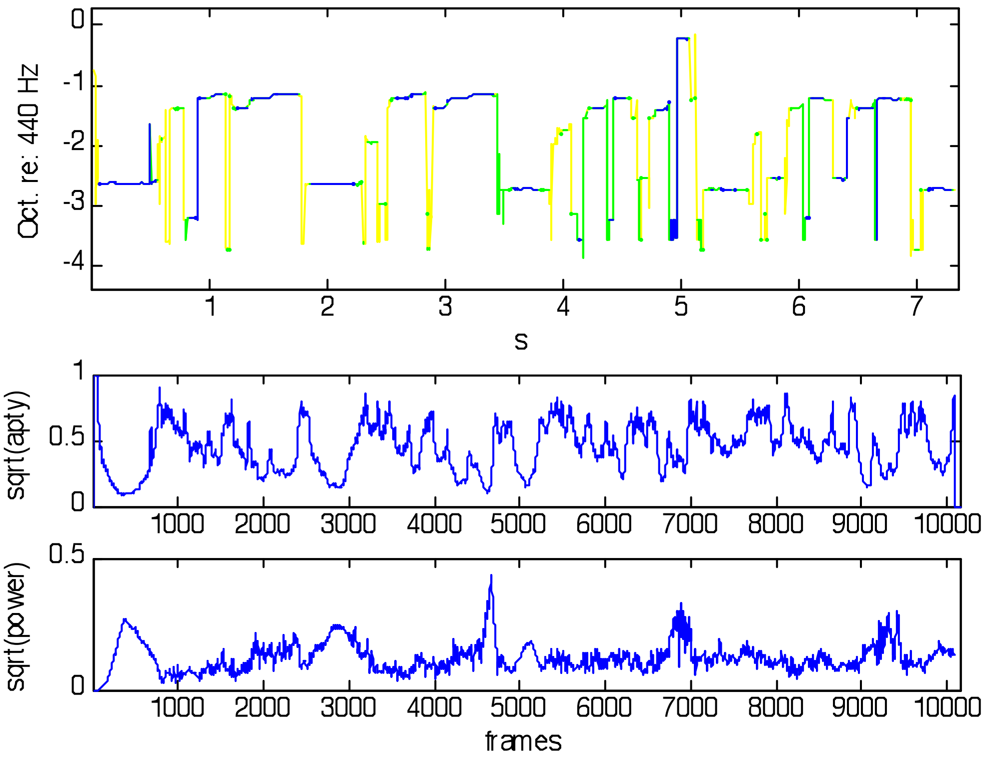
melody1.wav (brass)



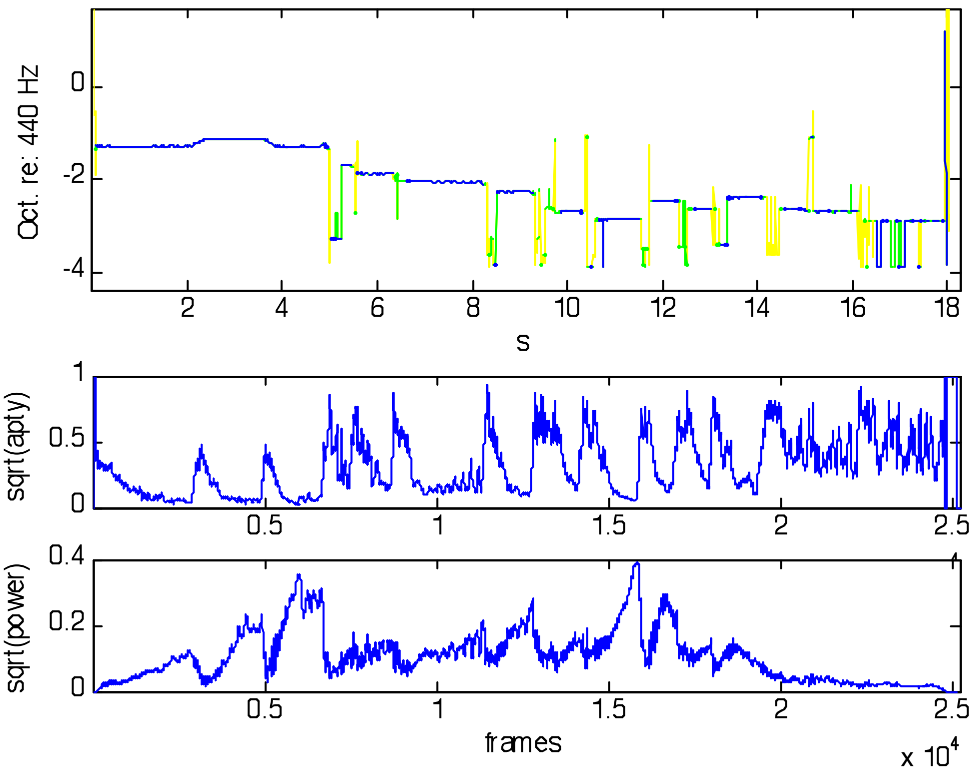
melody2.wav (bowed string)



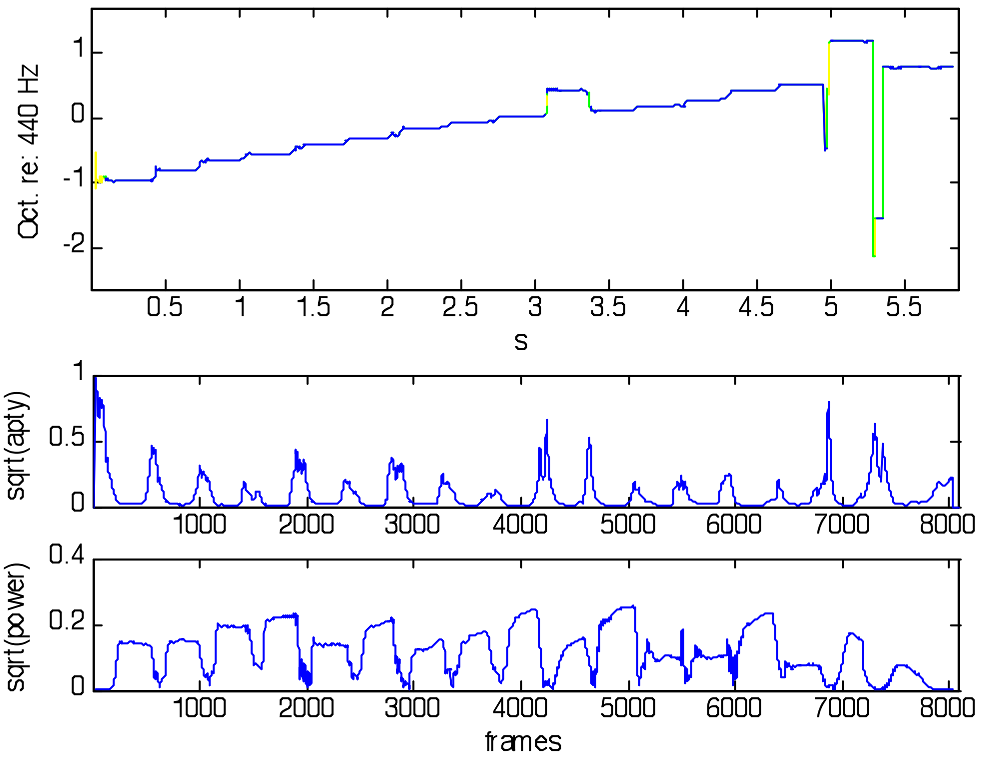
melody10.wav (bowed string)



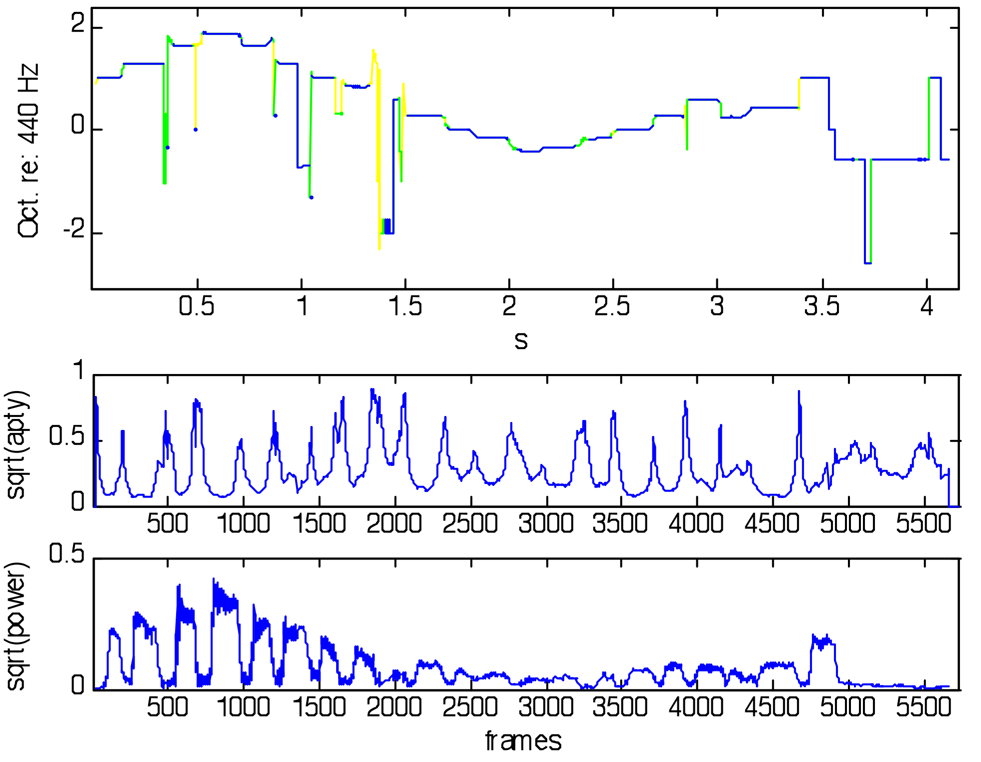
melody11.wav (bowed string, low pitch)



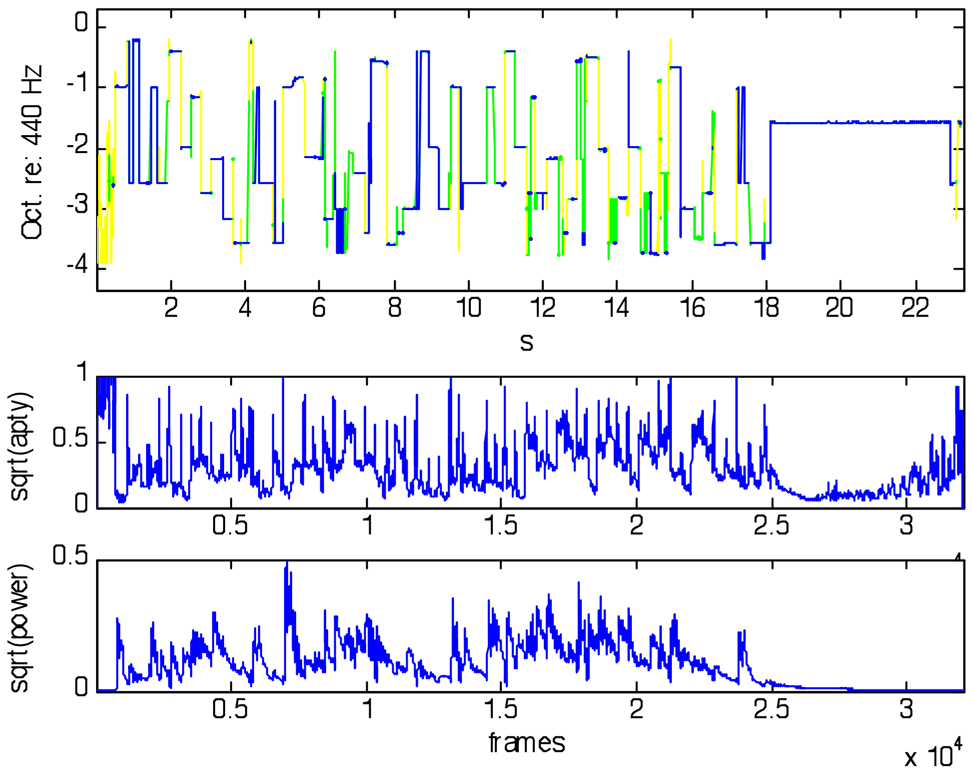
melody12.wav (wood wind)



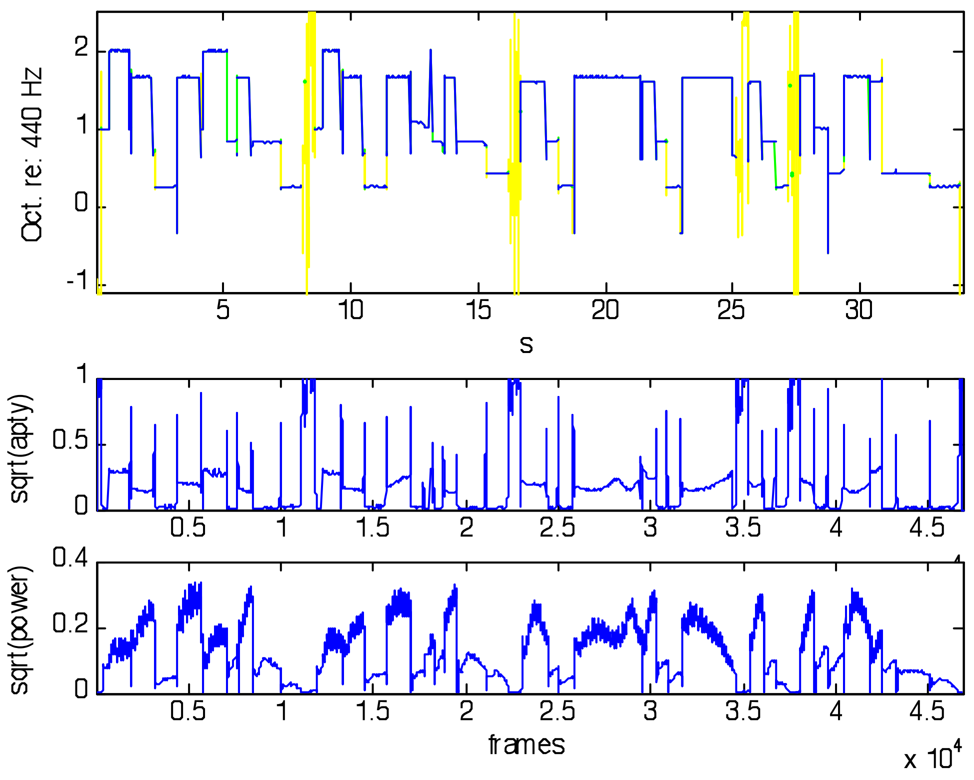
melody13.wav (flute)



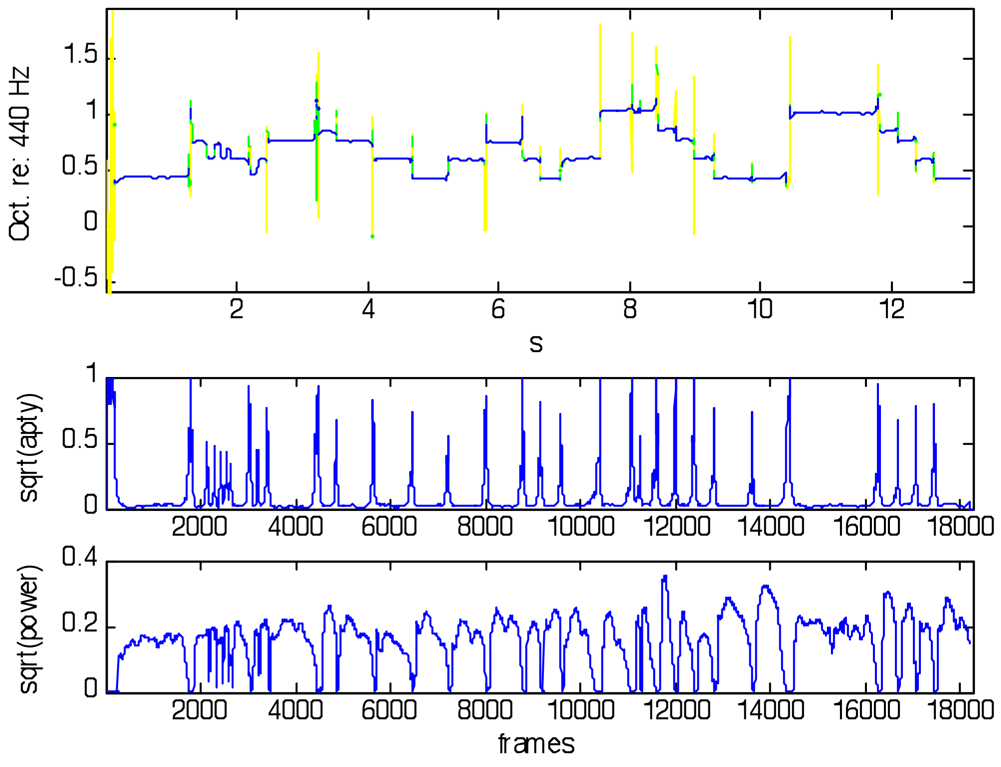
melody14.wav (guitar)



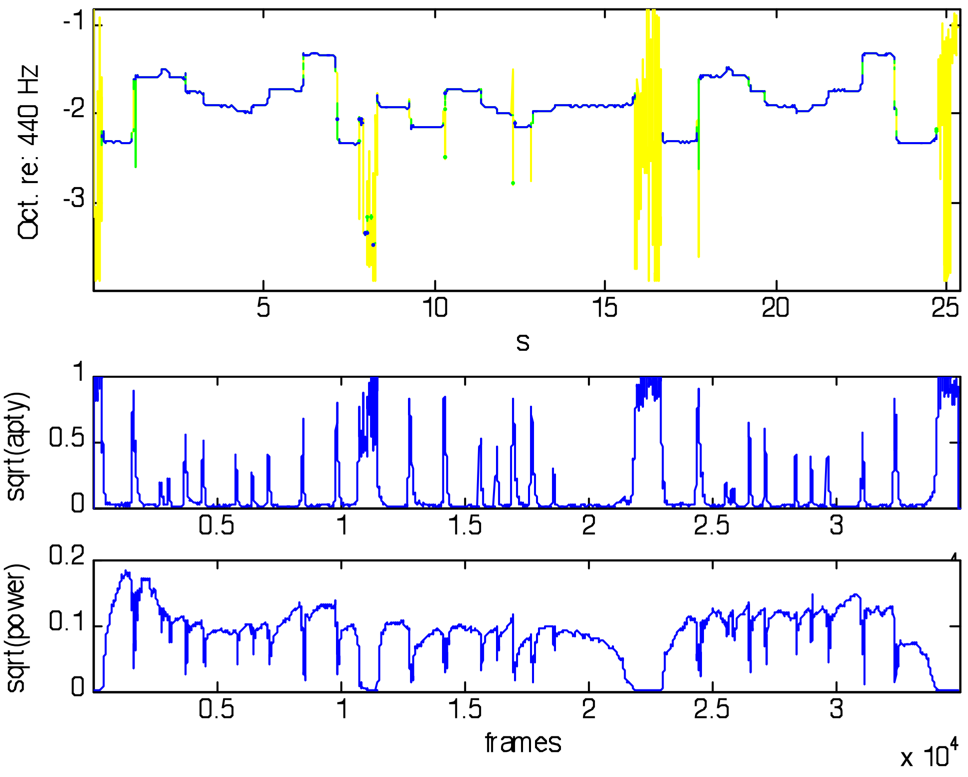
melody15.wav



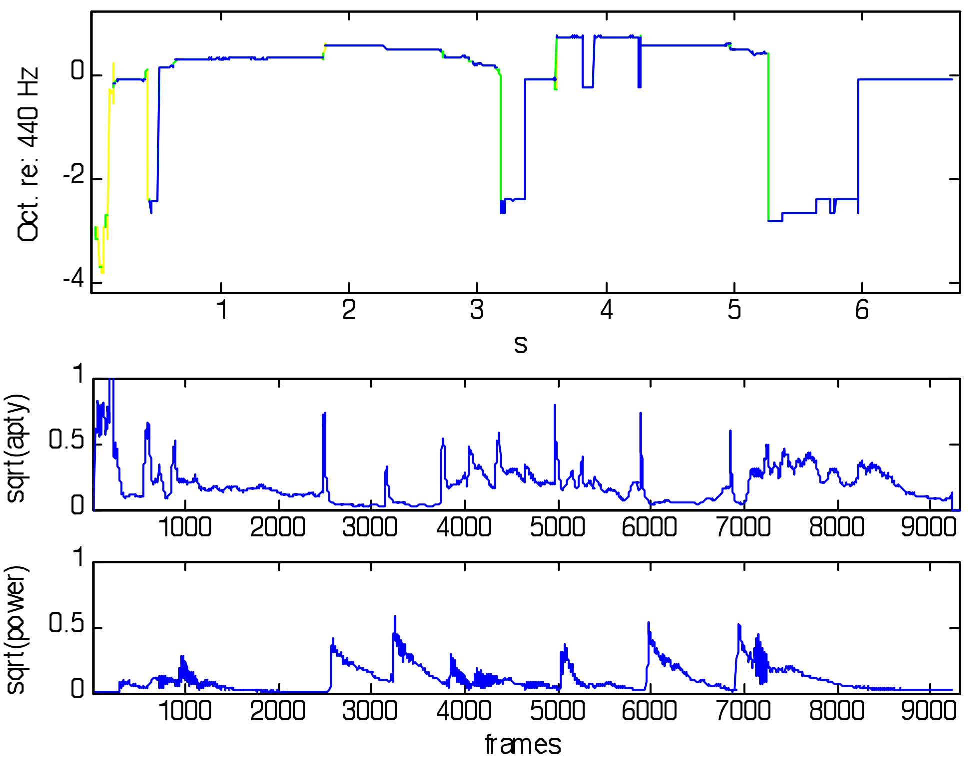
melody16.wav



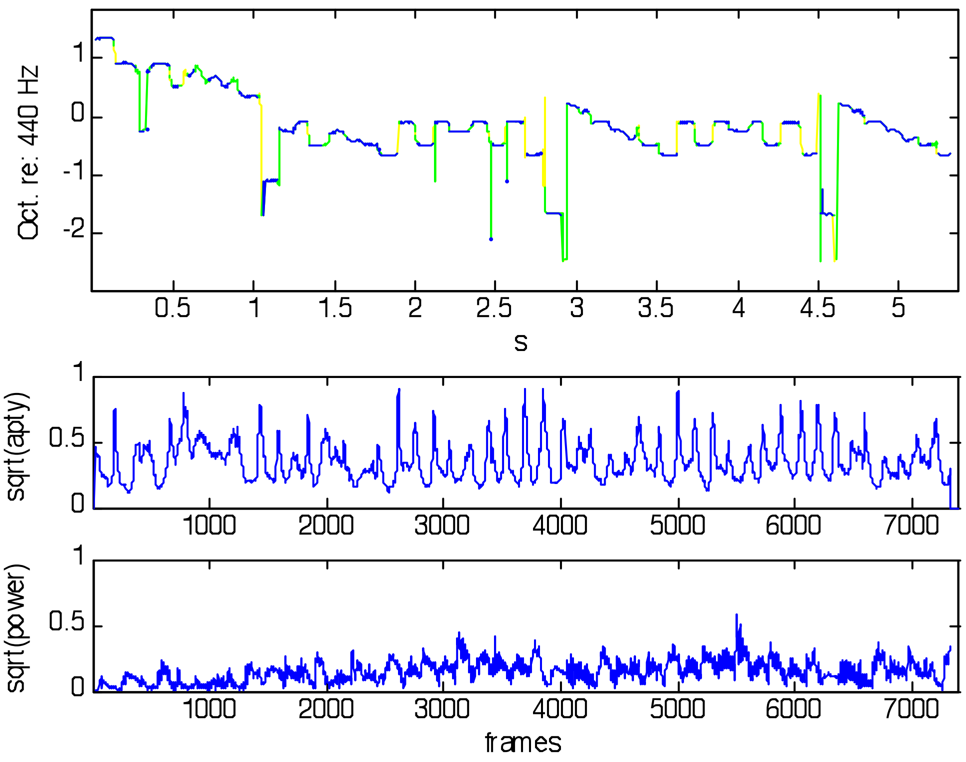
melody17.wav



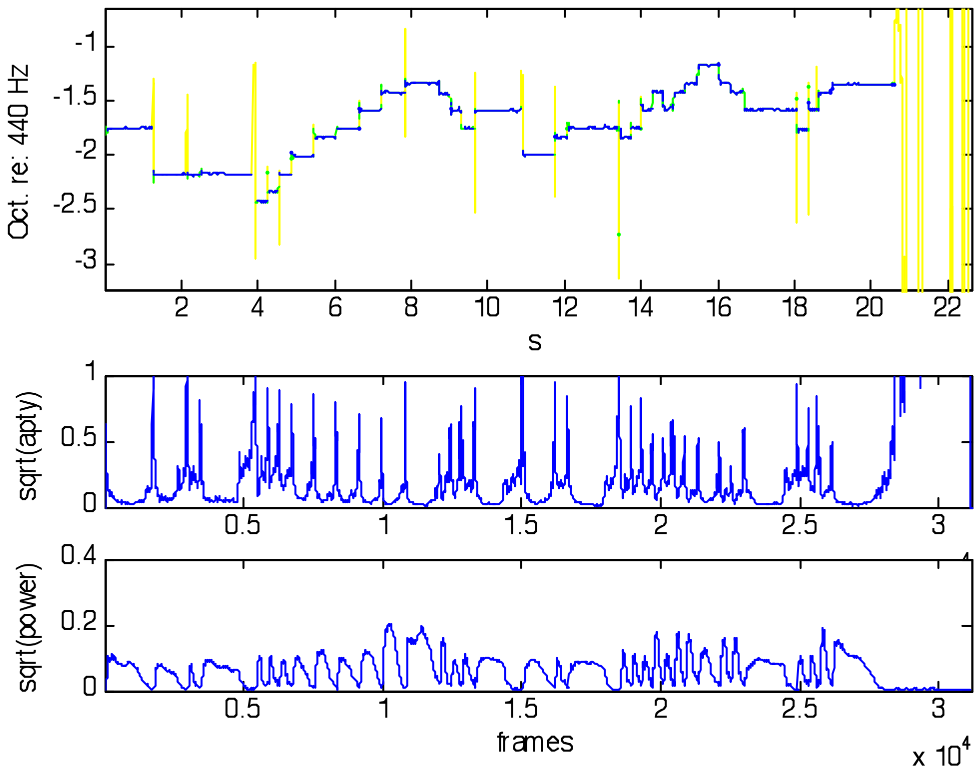
melody18.wav (piano)



melody19.wav (piano)



melody21.wav


Oct. re: 440 Hz
s
sqrt(apty)
sqrt(power)
frames
x 10^4