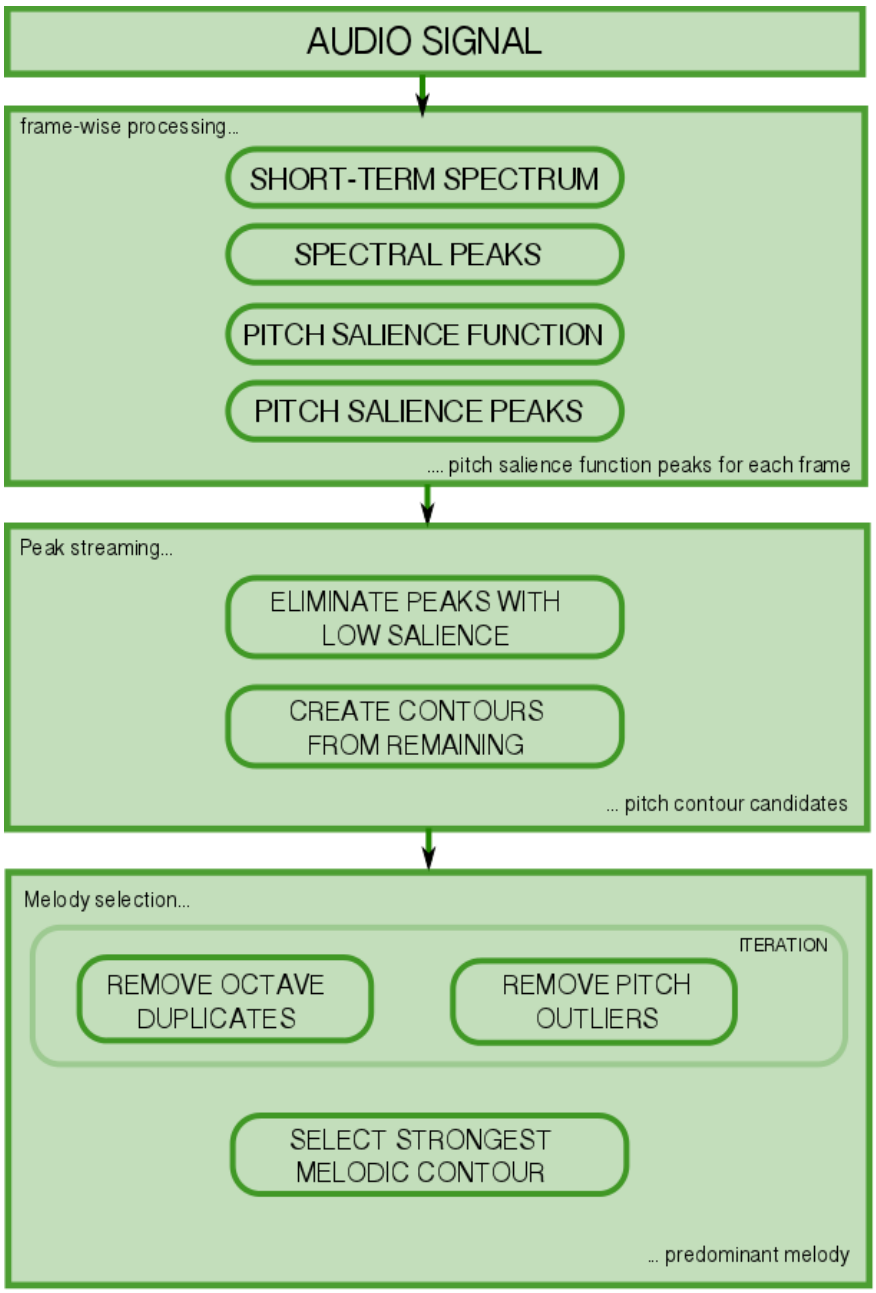
Unveiling the mystery: How the hell does MELODIA work?

The MELODIA algorithm estimates the predominant melody from polyphonic recordings. The challenge in the pitch estimation lies in the presence of various fundamental frequencies and their harmonics originating from multiple sources. We therefore use perceptual principles and assume that the fundamental of the predominant melody and its harmonics are stronger in magnitude and the pitch contour is furthermore continuous in time, i.e. does not include rapid changes or "jumps" in pitch.



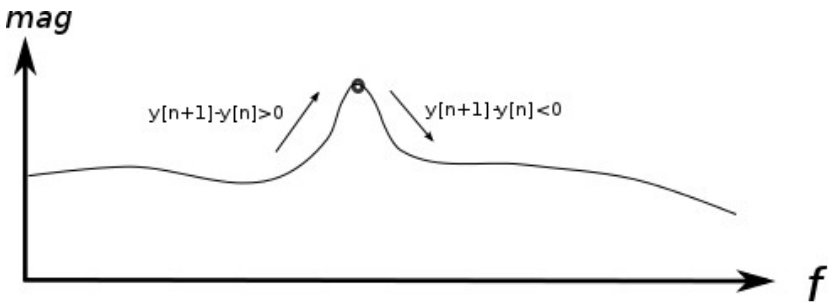
The algorithm can be divided into three main stages: First, in a pitch salience function is estimated on a frame-basis, gives for each frequency its strength as a potential fundamental. Its peaks are selected as melody candidates. In a second step, these peaks are grouped into melodic contours, according to their temporal and pitch continuity. Finally, the main melody is selected in filtering process, where pitch contours are eliminated in several stages. Here again, we take time and pitch continuity as well as perceptual salience (magnitude) into consideration.

<u>STAGE I:</u> Compute the pitch salience function and detect its peaks

Spectral peaks:

With default parameters, MELODIA segments into frames of 2048 samples with a hop size of 128. For each frame, we first compute the spectrum and determine its 100 strongest peaks. ESSENTIA uses a peak detection algorithm based on:

http://ccrma.stanford.edu/~jos/parshl/Peak_Detection_Steps_3.html

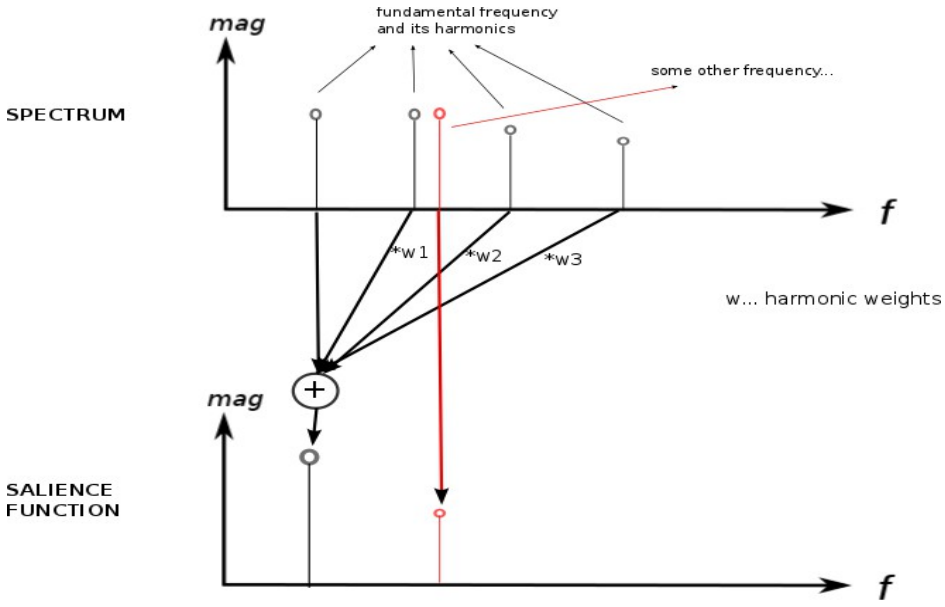


We "walk" through the spectrum from low to high frequencies. A peak will be characterized by the fact that we will walk "uphill" before and "downhill" after the peak. The algorithm therefore searches for changes in sign of the difference between consecutive magnitude values.

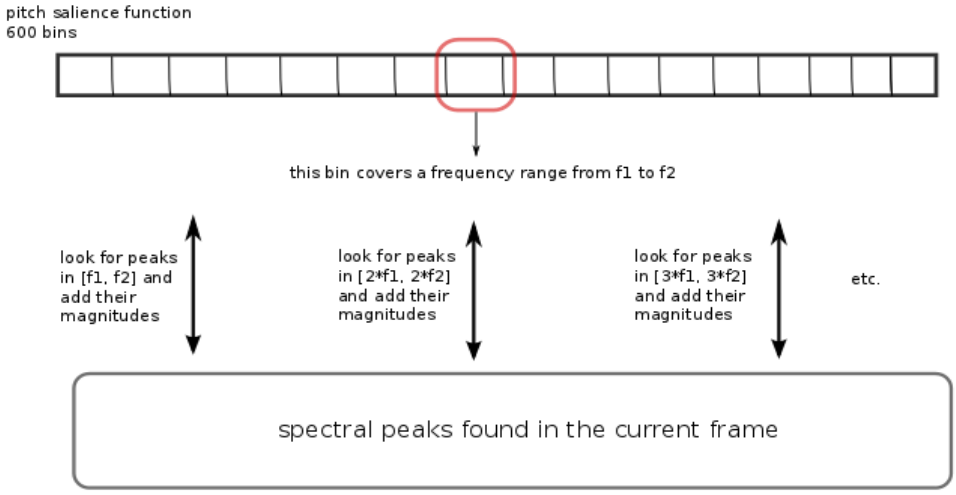
IDEA: Use a minimum peak distance (in frequency) of i.e. 50 cents to avoid giving to much weight to frequency ranges with resonances etc...

Pitch salience function:

We now compute the pitch salience function for each frame. For 600 cent-spaced bins, it contains a salience value, which indicates the possibility of this frequency bin being the fundamental in the current frame. We assume that the fundamental frequency of the main melody and its harmonics will have a strong magnitude.



For each of the bins in the salience function, we therefore determine if one of the spectral peaks is in this bin and if so, we add its magnitude. The same is done for the harmonics: For the first 20 harmonics, we determine if they are among the spectral peaks found and if so, we add them their magnitudes to the bin. MELODIA furthermore uses a weighting scheme, giving lower weights to higher harmonics and searches for spectral peaks in a certain range of +/- 50 cents around the actual frequency where again peaks further away from the centre frequency obtain a lower weight. To avoid adding magnitude values resulting from noise, magnitudes below a threshold are ignored.



The ESSENTIA implementation iterates over the found peaks and not the bins of salience function. It works more or less like this:

FOR (number of found peaks)
find the salience bin corresponding to the peak frequency
add the magnitude
FOR (number harmonics)
"imagine" the peak is the n-th harmonic, find the salience bin corresponding to f0
*add the magnitude * weight for n-th harmonic*
END
END

We end up with a pitch salience function for each frame containing 600 bins. A high salience value means that the corresponding centre frequency and its harmonics have strong magnitudes and the centre frequency is therefore more likely to be the fundamental frequency of the main melody. In order to obtain a discrete set of candidates, we again detect the peaks of the pitch salience function as described above.

At the end of this stage, we have a set of salience function peaks (max. 100) for each frame.

STAGE II: Peak streaming
[pitchcontours.cpp]

The idea is to group the pitch salience function peaks into contours which are continuous in salience value, pitch and time.

Classify into salient and non-salient peaks:

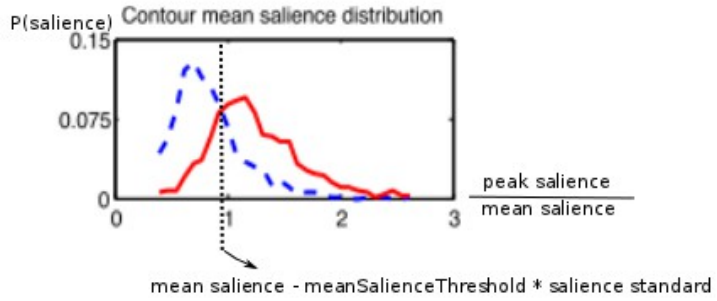
First, we find peaks which are low compared to the highest peak in the frame. The *frameSalienceThreshold* is an adjustable parameter.

Peaks which do not fulfil the condition

$$magnitude > frameSalienceThreshold * max(magnitudes\ in\ current\ frame)$$

are classified as non-salient. They are **not** eliminated, but saved for a later step.

We analyse the remaining peaks according to statistics calculated from the pitch salience function in the entire excerpt. First, we need to calculate the average salience value (magnitude) and its standard deviation of all peaks classified as salient. Peaks with a low salience compared to the average salience and are more likely to be non-melody peaks. We need to compare to the average salience over all frames, since the absolute value depends on the overall volume of the song and the spectral characteristics of the lead instrument.



red: melody peaks; blue: non-melody peaks

Peaks which do not fulfil the condition

$$salience > meanSalience - meanSalienceThreshold*salienceStandardDeviation$$

are classified as non-salient.

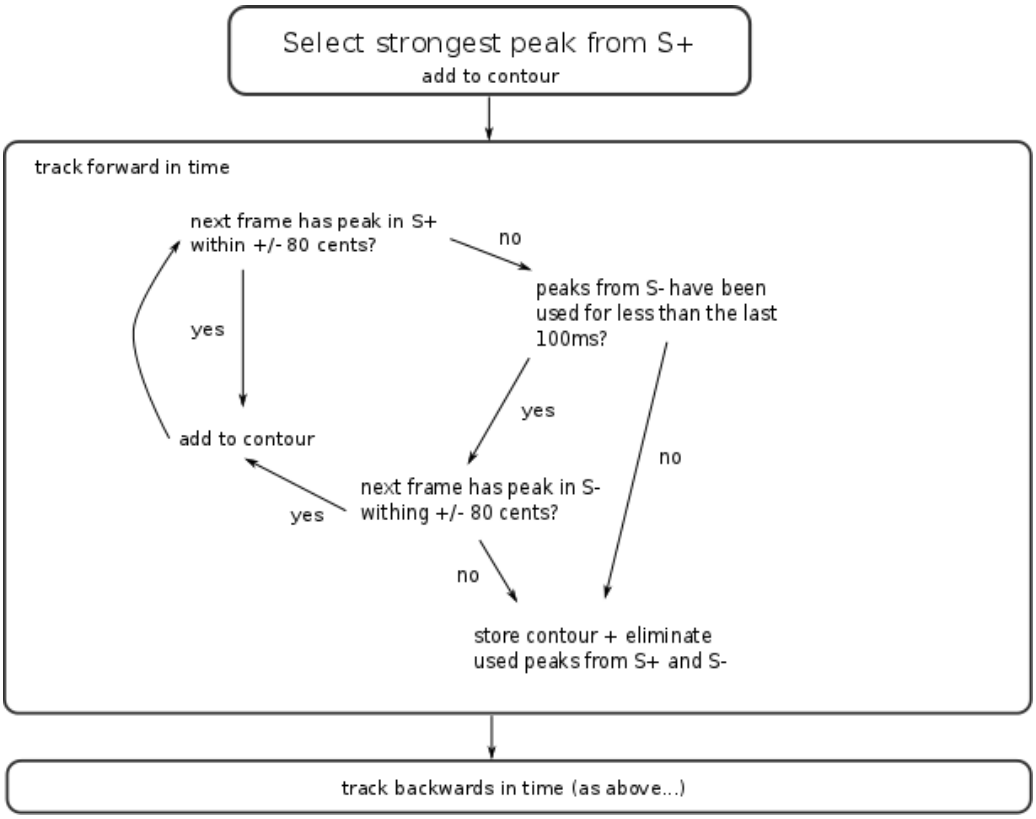
The *meanSalienceThreshold* is an adjustable parameter. This is of course a trade-of and high value will allow more false positives while low values will classify more melody peaks as non-salient. Also, this is not the voicing detection! We do not eliminate any peaks, we only classify.

We now have two sets of peaks, salient (S+) and non-salient (S-).

Create pitch contours:

We have already implemented the criteria that the main melody is perceptually dominant (it has a

strong f0 and strong harmonics) and now try to find contours which are continuous in pitch and time (“auditory streaming cues”).



We start with the strongest peak in S+ in the entire excerpt and track forward in time, by searching for peaks in the consecutive frame which are within 80 cents of pitch distance. The algorithm allows to include peaks from S- for a time span of max. 100ms, since the salience of the main melody might be temporally lower (loud cymbal strikes, noise, etc.). Once no more peaks are found, we go back to the initial peak and repeat the tracking, but backwards in time. The resulting contour is saved as a candidate for the main melodic line and the included peaks are removed from S+ and S- (a peak can only belong to one contour). The we look again for the highest peak in S+ and repeat the process until no more peaks are found. Short contours below a minimum duration threshold are discarded.

<u>STAGE III:</u> Melody selection
[pitchcontoursmelody.cpp]

We now have a set of pitch contours and want to determine, which belong to the main melody. Non-melody contours are usually associated to two causes: Octave errors and contours in non-melody (unvoiced) sections. We therefore first perform a voicing detection based on contour and salience characteristics. Then, pitch outliers and octave duplicates are detected and eliminated.

Voicing detection:

Analysing the statistics of contour characteristics of melody and non-melody contours reveals certain characteristics typical for contours belonging to the main melody: They have a high salience value, a large pitch standard deviation and often contain vibrato (singing voice is often the main element). While the salience can be seen as a universal parameter, the latter two criteria do not apply to all instruments carrying the main melody and are therefore used as and "immunity" parameter as explained later.

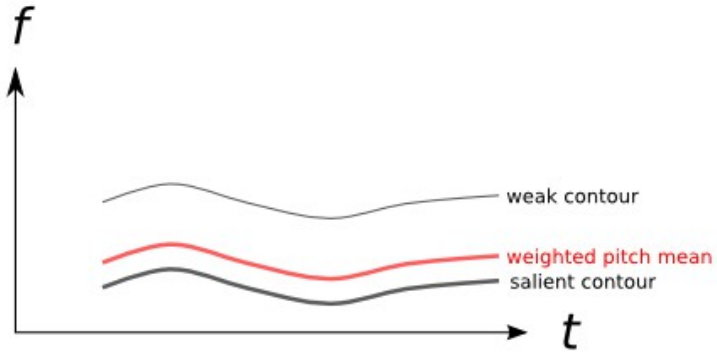
In a previous step we already eliminated frames with a low pitch salience compared to the mean salience. We will repeat this process, but now working on a contour level: We compare the mean pitch salience of a contour to the mean of the mean pitch salience over all estimated contours. Similar to the previous stage, contours are eliminated, which do not fulfil the following condition:

*meanContourSalience > mean(meanContourSalience) – contourSalienceThreshold*salienceStandardDeviation*

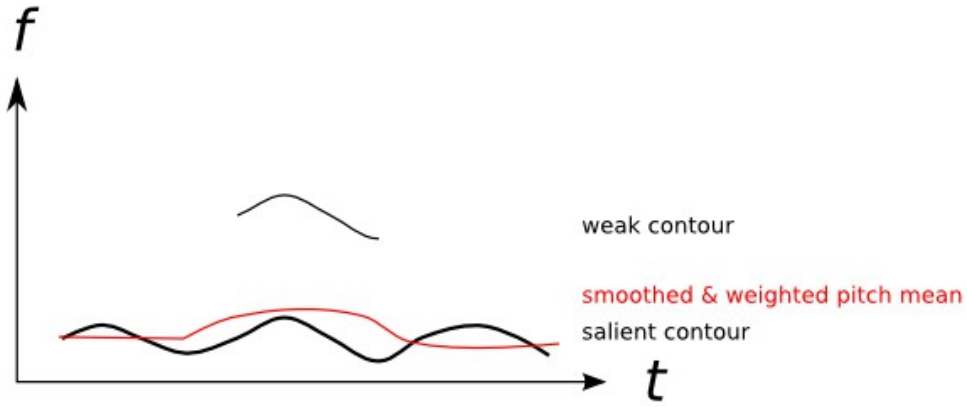
We give "immunity" to contours with a large pitch standard deviation or which contain vibrato, i.e. we do not eliminate them, even if they do not fulfil the condition stated above. Vibrato is detected by analysing the Fourier transform of the pitch contour – it will show as an isolated peak between 5Hz and 8Hz.

Contour duplicates and pitch outliers:

Octave errors in pitch estimation cause contour duplicates in octave distance. We can detect them by calculating the mean pitch distance of overlapping contours. For octave duplicates, the mean pitch distance will be within a range of 1200+/-50 cents. Now we need to decide, which of the contours belong to the main melody. We again refer to the assumption that melody contours have a stronger salience and are pitch continuous over time. This can be implemented by calculating the weighted pitch mean for the entire contour: For each frame, the pitch mean is given as the average value of the present contour frequencies weighted by their respective saliences. This weighting already produces a bias of the mean towards the contour with the higher salience:



We furthermore smooth the pitch mean with a 5 second moving average filter. In case of temporarily occurring contour duplicates, the surrounding contours will “pull” the mean towards the contour shows more pitch continuity in time.



The resulting pitch contour mean will accordingly be closer in pitch to contours which are more salient and show pitch continuity with surrounding contour segments. Octave duplicates can be eliminated by selecting the contour closer to the pitch mean and outliers are detected from their deviation to the pitch mean. In MELODIA, contour duplicates and pitch outliers are eliminated based on this technique in an iterative process with and adjustable number of filter iterations:

1) Calculate $\overline{P(t)}$ at each frame as the weighted mean of the pitch of all contours present in the frame.
2) Smooth $\overline{P(t)}$ using a 5-second sliding mean filter (length determined empirically) with a hop size of 1 frame. This limits the rate at which the melody pitch trajectory can change, ensuring continuity and avoiding large jumps.
3) Detect pairs of octave duplicates and, for each pair, remove the contour furthest from $\overline{P(t)}$.
4) Recompute $\overline{P(t)}$ using the remaining contours, following Steps 1-2.
5) Remove pitch outliers by deleting contours at a distance of more than one octave from $\overline{P(t)}$.
6) Recompute $\overline{P(t)}$ using the remaining contours, following Steps 1-2.
7) Repeat Steps 3-6 twice more, each time starting with all contours that passed the voicing detection stage, but using the most recently computed melody pitch mean $\overline{P(t)}$. The number of iterations was chosen following experimentation suggesting this was sufficient for obtaining a good approximation of the true trajectory of the melody. In the future we intend to replace the fixed iteration number by a stabilisation criterion.
8) Pass the contours remaining after the last iteration to the final melody selection stage.

Final melody selection:

The entire process usually leaves only a single peak at a time frame. In case a frame still contains more than one contour, the frequency with the higher salience is chosen.