Efficient adaptive multiresolution representation of music signals

Nícolas Silvério Figueiredo

Dissertation presented to the
Institute of Mathematics and Statistics
of the University of São Paulo
in partial fulfillment
of the requirements
for the degree of
Master of Science

Program: Computer Science

Advisor: Prof. Dr. Marcelo Queiroz

During this work, the author received financial support from CAPES

São Paulo, December 14th, 2020

Efficient adaptive multiresolution representation of music signals

This version of the thesis includes the corrections and modifications suggested by the Examining Committee during the defense of the original version of the work, which took place on December 14th, 2020.

A copy of the original version is available at the Institute of Mathematics and Statistics of the University of São Paulo.

Examining Committee:

- Prof. Dr. Marcelo Gomes de Queiroz (advisor) - IME-USP
- Prof. Dr. Emmanouil Benetos - EECS-QMUL
- Prof. Dr. Tiago Tavares - NICS-UNICAMP

Resumo

FIGUEIREDO, N. S. **Representação eficiente adaptativa multiresolução de sinais musicais**. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

A inerente troca entre resolução no tempo e na frequência de transformadas convencionais (como a Transformada Discreta de Fourier) pode ser um inconveniente na representação de sinais musicais, já que tais transformadas são incapazes de localizar simultaneamente eventos percussivos com precisão no tempo e eventos melódicos com precisão na frequência. Representações adaptativas buscam contornar essa limitação variando o tamanho da janela de análise utilizada em cada região do plano tempo-frequência, e possuem aplicações como entrada para algoritmos automáticos de transcrição de música, separação de fontes e análise de expressividade musical.

O projeto apresentado tem como objetivo principal o desenvolvimento de uma representação adaptativa de baixo custo computacional, cuja estrutura se opõe à tradicional combinação de representações de diferentes resoluções pré-computadas. O proposto Iteratively Refined Multiresolution Spectrogram (IRMS) funciona a partir de refinamentos sucessivos em cima de um espectrograma inicial de baixa resolução de frequência, localizados nas áreas do plano tempo-frequência nas quais existe informação musical como notas, harmônicos e elementos expressivos. Seu desenvolvimento passa pela investigação de estimadores de informação musical e técnicas de processamento em sub-bandas que permitam uma computação eficiente de representações em alta resolução de regiões isoladas do plano tempo-frequência.

Para a investigação de algoritmos de processamento em sub-bandas para essa finalidade, foi desenvolvida uma aplicação que permite a visualização em alta resolução de áreas específicas de um espectrograma. Um experimento comparativo entre diferentes estimadores de informação musical foi conduzido, com bons resultados para as entropias de Shannon e Rényi. Também são apresentados detalhes técnicos sobre a integração entre detecção de subregiões musicais e seu refinamento via processamento em sub-bandas, que dá origem à implementação final da IRMS. Como avaliação da solução, um experimento final comparativo baseado em custo computacional entre diferentes representações no plano tempo-frequência foi realizado. A IRMS alcançou tempos de execução ordens de magnitude menor do que as outras representações adaptativas avaliadas, e em algumas configurações apresentou custo computacional competitivo em relação à CQT e à STFT, validando a nossa proposta de uma alternativa eficiente para representações adaptativas.

Palavras-chave: representação multi-resolução, representação adaptativa, transcrição automática de música, computação sonora e musical.

Abstract

FIGUEIREDO, N. S. **Efficient adaptive multiresolution representation of music signals**. Dissertation (Masters degree) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2020.

The inherent trade-off between time and frequency resolutions, which exists in conventional transforms (such as the Discrete Fourier Transform) may be a hindrance for the representation of music signals, since these transforms are incapable of simultaneously locating percussive events with precision in time and melodic events with precision in frequency. Adaptive representations intend to address this limitation by varying the analysis window size used in sub-regions of the time-frequency plane (TFP), and can be used as input representations in algorithms for automatic music transcription, source separation and musical expressiveness analysis.

The main objective of the presented work is the development of an efficient adaptive transform, that serves as a counterpoint to traditional algorithms based on the combination of precomputed representations with different resolutions. The proposed Iteratively Refined Multiresolution Spectrogram (IRMS) works by performing successive refinements on top of an initial low frequency resolution spectrogram, located in the areas of the TFP that contain musical information such as notes, harmonics and expressive elements. Its development is built on the investigation of musical information estimators and sub-band processing techniques that allow the efficient computation of high resolution representations within isolated subregions of the TFP.

As an investigation of sub-band processing algorithms for this task, a GUI application was built for the detailed high-resolution visualization of specific areas of a spectrogram. A comparative experiment between different musical information estimators was conducted, with good results for Shannon and Rényi entropies. This work also presents technical details on the integration between the detection of musically relevant subregions and their refinement via sub-band processing, that defines our final implementation of the IRMS. As an evaluation of the final solution, a comparative experiment based on computing cost between different time-frequency representations was conducted. The IRMS achieved execution times orders of magnitude faster than the other evaluated adaptive representations, and in some configurations presented a competitive computational cost with respect to the STFT and CQT, thus validating our proposal of an efficient alternative for adaptive representations.

Keywords: multiresolution representation, adaptive representation, automatic music transcription, sound and music computing.

Contents

Chapter 1

Introduction

1.1 Context

Frequency domain representations are an integral part of signal processing, whether it be for the analysis of speech, image and music signals, or of other signals in areas such as telecommunications, medical sciences, applied mathematics, physics and many others. Specifically in the Sound and Music Computing research area, tasks like source separation, automatic music transcription, music information retrieval (MIR), signal transformation and noise reduction generally use as starting point frequency domain representations of the analyzed signals. Consequently, all of the processing and its results are conditioned by the quality of this representation, which, if computed with a non-optimal resolution, can lead to blurring and poor localization of events in the time-frequency plane.

Recently, with the rise of machine learning techniques in the MIR area, large data sets have become an essential part of research. These data sets are mostly comprised of music (annotated or not) in their time domain representation, and algorithms that use frequency representations as input data must compute them for the whole data set, which puts focus on the computing cost of these analysis techniques. This landscape serves as context and motivation for the investigation of alternative adaptive algorithms for frequency domain representations of music signals.

1.2 Motivation and applications

The spectrogram is a widely used analysis/representation tool in sound and music computing, despite the fact that the linear frequency resolution of the Short-Time Fourier Transform (STFT) is rarely adequate for applications dealing with music signals. Pitches in the 12-tone tempered scale (which is dominant in the western music) are distributed in a logarithmic fashion: the so-called semitone distance between adjacent tones is always relative and roughly equal to 5.9463% of the frequency of the lowest tone (see Fig.1.1). Furthermore, the human ear presents an approximately exponential resolution between 500 Hz and 20 kHz [Moo95], making it capable of more easily differentiating small frequency changes in the lower registers as opposed to higher registers. This serves as motivation for representations that sample different frequency ranges at different rates, which are called **multiresolution representations**.

The trade-off between frequency and time resolution, which is a defining characteristic of the STFT (discussed in detail in Section 2.2), also serves as motivation for the use of other representations besides the STFT spectrogram. This trade-off, also referred to as the uncertainty principle, makes it impossible for signals with melodic and percussive spectral characteristics such as music signals to be sparsely represented by a single STFT analysis, since either percussive events or melodic events will be poorly located in the time-frequency plane according to the analysis window chosen. This causes problems for tasks such as automatic music transcription (AMT) that depend on the precise detection of note onset times and melody extraction. Analysis of musical expressiveness (the automatic identification of *glissandi*, tremolo, vibrato) also depends on both precise time

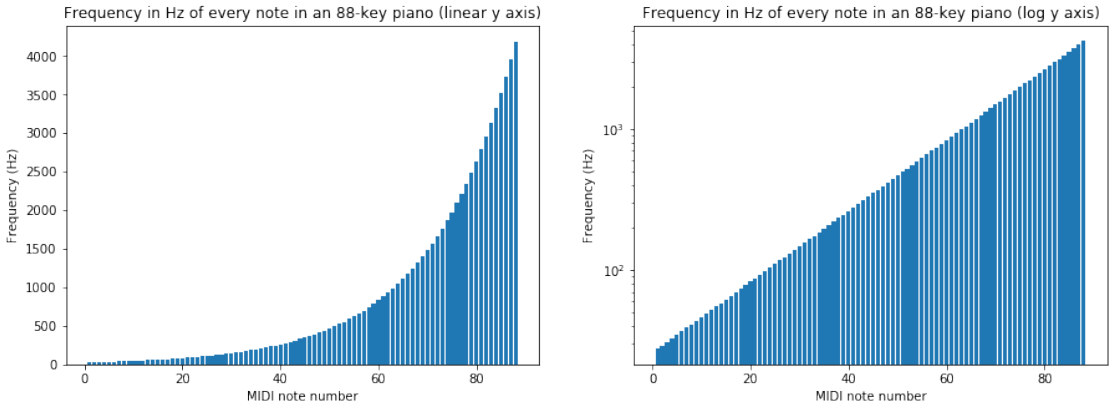


Figure 1.1: *The frequency in Hz of every note in an 88-key piano, with a linear and logarithmic y axis: the linear aspect of the logarithmic plot illustrates the logarithmic distribution of frequencies for the 12-tone tempered scale.*

and precise frequency resolutions. These tasks motivate the development of **adaptive representations** for music signals, which are representations that vary both time and frequency resolutions in different regions of the time-frequency plane, according to local characteristics of the signal being analyzed.

Another motivation for the research proposed here arises from a prevalence found in the literature on adaptive representations: most algorithms follow the same framework, where several representations are precomputed for the entire signal, and then used to compose a multiresolution spectrogram. This introduces an inefficiency in the process: several regions of these representations are discarded after their computation, causing an enormous computational overhead. This motivates the investigation of a more efficient representation that avoids the unnecessary computation of high-resolution regions of the time-frequency plane, performing them only when there is evidence of the presence of relevant musical events in that region. This way, such a representation could eventually use the same amount of data of a fixed-resolution spectrogram while providing a more precise representation, by using very coarse resolutions in areas that do not contain musical events and high resolution representations of musically relevant regions.

The applications of such a representation are many: as stated, this would provide a useful representation for AMT-related tasks such as melody extraction, onset detection and musical expressiveness analysis. Given that the adaptive strategy aims to precisely locate all relevant musical events present in a signal, this is a potentially valuable representation for source separation algorithms, because the amount of noise and cross-components between sources would be minimized. In more general terms, since its objective is to produce an accurate representation of a music signal, it could potentially serve as an alternative input representation for any MIR algorithm that relies on spectrograms as input data.

1.3 Related work

Among multiresolution representations, the Constant-Q Transform (CQT) [Bro91] is widely used as an input representation for tasks such as monophonic fundamental frequency tracking [Bro92], source separation [JFB+11, GSD12], and pitch shifting [SKS13]. Since the main motivation behind the CQT is the representation of music signals, it takes into consideration the logarithmic distribution of frequency of the notes in the tempered scale. Because of that, its frequency resolution is geometrically related to the absolute frequency value: the Q factor is given by $Q = f/\delta f$ where f is the frequency value and δf is the resolution, and it is constant for every value of f. This results in each musical octave being represented with the same number of bins. In [SKHD14], an expansion of the CQT called the Variable-Q Transform is presented, where the Q factor is allowed to vary smoothly in order to create constant filter bandwidths towards the lower frequencies. This

is done in order to mitigate the cost of using very long windows in lower octaves, and to make its filters more closely resemble the human auditory system, which is approximately constant-Q only for frequencies above 500 Hz and approaches constant bandwidth in lower frequencies.

A general framework for adaptive representations is presented in [LT06], along with a specific algorithm. In it, the time-frequency plane is divided into sub-regions and for each one the sparsest representation amongst a set of alternatives is chosen to represent it; the sparsity measure chosen is a form of entropy. By effectively varying the window size used in order to produce the representation of each subregion, this solution tries to mitigate the uncertainty principle by maximizing the sparsity of the final representation. In [dC20], several methods for the combination of previously computed spectrograms are presented and divided into three groups: bin-wise combinations, combinations based on local information and combinations based on image analysis. Similarly to the work in [LT06], this second group presents methods that utilize the Gini index as a measure of sparsity used to weight the combination of the precomputed spectrograms. A formal mathematical framework for the analysis, transformation and resynthesis of a signal with adaptive time-frequency resolution based on nonstationary Gabor frames is developed in [LRM+13]. Rényi entropies are used as a sparsity measure for the choice between different sets of analysis windows at each time frame of a signal, and a resynthesis method is provided along with a theoretical upper bound for its error. A similar algorithm is given in [JT07]: the time-frequency plane is divided into rectangular regions, and for each one the sparsest (according to Rényi entropy) Gabor representation among a family of representations is chosen. Then, this initial representation is inverted, and subtracted from the original signal. The resulting residual signal is again approximated using the same adaptive algorithm, and the process is iterated until a criterion is met, resulting in a layered representation of the original signal. Besides entropy, energy variance has also been used as a measure of relevance: in [ZN77], an adaptive coding technique for speech signals is developed using energy variance of the transform coefficients as an information estimator.

Other relevant multiresolution representations and methods include discrete wavelet transforms, which represent a given signal using basic functions derived from a mother wavelet that can be of varying frequency and limited time duration. Dyadic [FS99] and Multiplexed [Eva93] wavelet transforms have been investigated as the basis for pitch estimation algorithms. Reassignment methods [HM03] use phase information in order to represent simple sinusoids, linear chirps, and impulses with a higher resolution than the inherent FFT resolution tradeoffs, and have been used for melody extraction [GH99]. The Coupled PLCA [NMG+10] method computes a multiresolution representation by jointly decomposing two spectrograms, one with a high frequency and one with a high temporal resolution. This method is applied successfully to speech analysis, although conversion of the algorithm is not guaranteed. While these are valid approaches for computing multiresolution representations, this work focuses on adaptive methods that use the STFT as their foundation in order to maintain a clear interpretability of results based on the basic waveforms of the FFT and musical information on the TFP.

Sub-band processing is found on many compression algorithms for audio, images and video, such as JPEG [Wal92], MUSICAM [DLU91] and MPEG [LG91]. In [TC79], a general sub-band coding framework is presented where the original signal is analyzed with a time-domain filter bank, and each band is ring-modulated to DC and low-passed before being subsampled. A more efficient sub-band coding technique is presented in detail in [VSW91]: the modulation, low-pass filtering and subsampling steps are substituted by a single *undersampling* step, that in effect subsamples and modulates the signal at the same time. In [Rot83], polyphase quadrature filters are used for a computationally efficient front-end for sub-band coding. These techniques are also a part of multiresolution transforms: in [SK10], an efficient algorithm for the computation of the CQT is presented, where CQT coefficients are computed one octave at a time, based on a recurring analysis, filtering and subsampling algorithm.

1.4 Objectives

As stated in Section 1.2, tasks such as automatic music transcription motivate the development of representations that accurately represent all musical events contained in a music signal. From the studied solutions, the additional motivation of exploring a more efficient approach to adaptive representations was formulated.

In summary, this research project has as its main objective the development of an efficient adaptive time-frequency representation based on the detection of musically relevant subregions of a spectrogram and their iterative refinement. This development is supported by the investigation of the following questions:

- How reliably can we detect the subregions of a spectrogram that contain musical events?
- Is sub-band processing an appropriate solution for the efficient refinement of isolated regions of the time-frequency plane? What would be a suitable sub-band processing algorithm for this task?
- Is the proposed scheme of detection and refinement computationally more efficient than the traditional framework of combination of precomputed representations? Additionally, how does it compare, with respect to computing costs, to other single resolution and multiresolution representations?

These questions are further detailed in Chapter 3.

1.5 Structure of this Dissertation

Chapter 2 introduces the conceptual and technical tools involved in the development of this project, along with technical discussions of related work. Chapter 3 begins with a conceptual overview of our adaptive representation proposal. Further details are given into the questions raised by this proposal and the steps planned in order to investigate them and accomplish the objectives of this work. A technical description of the implementation of both a tool relating to the explored sub-band processing techniques and the proposed adaptive representation itself is given in Chapter 4. Chapter 5 lays out the technical details of the conducted experiments, along with their results and discussions. Finally, Chapter 6 offers an overview of this research project, its contributions and future work.

Chapter 2

Fundamental Concepts

This chapter introduces the conceptual tools which are central to the the development of this project and offers a technical description of related work.

2.1 STFT and spectrogram

2.1.1 Time and frequency domain representations of audio signals

Audio is represented digitally by sampling the sound pressure wave at a constant rate. This rate f_s is called the sampling rate, and its inverse $T_s = 1/f_s$ is the sampling period. Therefore, a digital audio file can be understood as a sequence of values $\{x_n\} \in R$ that describe a waveform over time. This is the **time domain** representation of an audio signal (see Figure 2.1a).

Signals can also be represented in the **frequency domain** by their decomposition into a sum of basic functions. The Fourier transform decomposes a signal into a sum of sinusoidal components. The Discrete Fourier Transform (DFT) of a signal $\{x_n\} = x_0, x_1, ..., x_{N-1}$ containing N samples is given by $\{X_k\} = X_0, X_1, ..., X_{N-1}$ where:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi ikn/N}, \tag{2.1}$$

and expresses the original signal as the following linear combination of complex exponential signals:

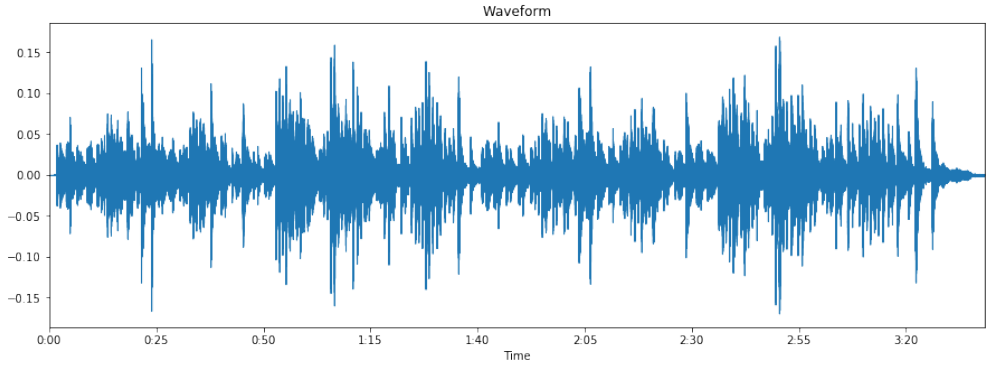
$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{2\pi ikn/N}. \tag{2.2}$$

2.1.2 Spectrogram

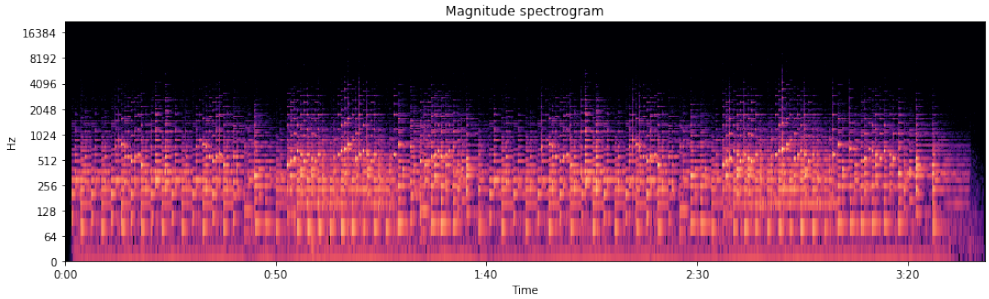
The Short Time Fourier Transform (STFT) performs the above decomposition repeatedly for small segments called **window frames** of a signal. The result is a matrix that represents the evolution of these coefficients over time. Since these coefficients are complex values, it is often more useful to plot their magnitudes (resulting in a **magnitude spectrogram**) or their squared magnitudes (resulting in a **power spectrogram**) as a heat map (see Figure 2.1b). The plane in which these values are plotted is called the **time-frequency plane (TFP)**. This type of visualization is widely used in the area of signal processing.

2.2 Time-frequency resolution

The formula for the IDFT in Eq. 2.2 expresses the original signal as a linear combination of complex exponential signals of form $e^{2\pi ikn/N}$. For a signal sampled at f_s Hz, the translation between



(a)



(b)

Figure 2.1: ***a)*** *Waveform representation of an audio signal (time domain representation)* ***b)*** *Magnitude spectrogram of an audio signal (frequency domain representation)*

time t in seconds and sample index n is given by $n = t \cdot f_s$, which means Eq. 2.2 can be rewritten as:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{2\pi i k t f_s / N}. \tag{2.3}$$

It is worth noting that the analog waveform $e^{2\pi i q t}$ makes q complete cycles per second, therefore the exponential signals being used to represent the original signal have frequencies equal to $k \cdot f_s/N$ Hz. From this, the following observations can be made:

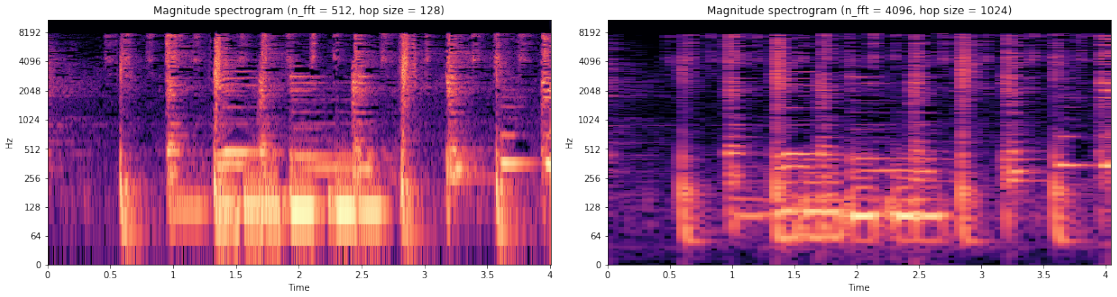
1. The DFT represents the analyzed signal by a linear combination of complex signals with frequencies evenly spaced by f_s/N Hz. In other words, the frequency resolution of the DFT is given by f_s/N Hz, where f_s is the sampling frequency and N is the size of the analysis window in samples;

2. The frequency resolution varies with the analysis frame size N and hence with the duration of the signal N/f_s.

Observation 1 states that the DFT is a fixed resolution transform. Since the frequencies of the basic waveforms used to represent the analyzed signal are linearly spaced, equal importance is given for each frequency band on a linear scale, and thus increasing importance is given to higher pitches (more coefficients per music interval). Observation 2 explains the trade-off between frequency and time resolution of the DFT, which also reflects the global nature of the basic waveforms[1]. Whenever a finer frequency resolution is needed, the analysis window size N needs to be bigger, which means that a bigger time slice of the signal has to be analyzed, and so the spectral characteristics of the adjacent portions of the original signal slice are now part of the same analysis coefficients (leading to

[1]These are global in the context of the analysis window, in the sense that each waveform $e^{2\pi i k t/T}$ in the IDFT Equation 2.2 represents an unchanging frequency component within the window.

a coarser time resolution). If a finer time resolution is needed, N needs to be smaller, thus lowering the number of frequencies sampled by the DFT (leading to a coarser frequency resolution). This trade-off makes it impossible for signals with both melodic and percussive spectral characteristics such as music signals to be adequately represented by an STFT analysis: as shown in Fig. 2.2, percussive hits will be poorly time-located if a big analysis window is used (right), and melodic lines will be poorly frequency-located if a small analysis window is used (left).



(a) **(b)**

Figure 2.2: *Two spectrograms of the same time slice of a music signal containing percussive and melodic elements.* ***a)*** *A high time resolution spectrogram (analysis window of 512 samples) that locates poorly the melodic events (particularly the note at 128 Hz) in the frequency axis.* ***b)*** *A high frequency resolution spectrogram (analysis window of 4096 samples) that locates poorly the percussive events in the time axis.*

2.2.1 Multiresolution representations

In order to circumvent this trade-off, multiresolution representations were developed. These are representations that do not use evenly-spaced sampled frequencies: more importance is given to specific frequency bands in detriment of others.

Among multiresolution representations, the Constant-Q Transform (CQT) [Bro91] is of special importance for this work. It is a transform developed for the analysis of music signals, motivated by the logarithmic distribution of the frequencies of notes in the tempered scale. Its frequency resolution is geometrically related to the frequency: there is a constant ratio between frequency value and resolution, given by the Q factor. For example, if a distinction between semitones is needed in the analysis, one often chooses a quarter-tone frequency resolution ($\delta f = 2^{\frac{1}{24}} - 1$), and so $Q = f/\delta f = f/(0.029f) = 34$ in this case.

This transform was initially presented in [Bro91] as an adaptation of the DFT formula, varying the analysis window size according to the frequency being sampled. This causes the analysis of lower frequencies to be computed with very large windows, resulting in high computational cost. For instance, if $Q = 34$, for a signal sampled at 48 kHz, the 100 Hz frequency is sampled with a window of size N=48000/(100/34)=16320.

2.2.2 Adaptive representations

Another way to address the trade-off issue is to vary the analysis window size according to some characteristic of the signal being analyzed. Ideally, for the analysis of music signals, small windows would be used in moments where percussive events occur, and large windows would be used in moments where melodic events occur. This way, percussive elements would be time-located with precision and melodic elements would be frequency-located with precision. Representations that adapt their characteristics according to the signal being analyzed are called **adaptive representations**.

Lukin & Todd [LT06] present a general framework for adaptive representations (depicted in Fig. 2.3), where several representations of a signal are processed in parallel using different filter banks

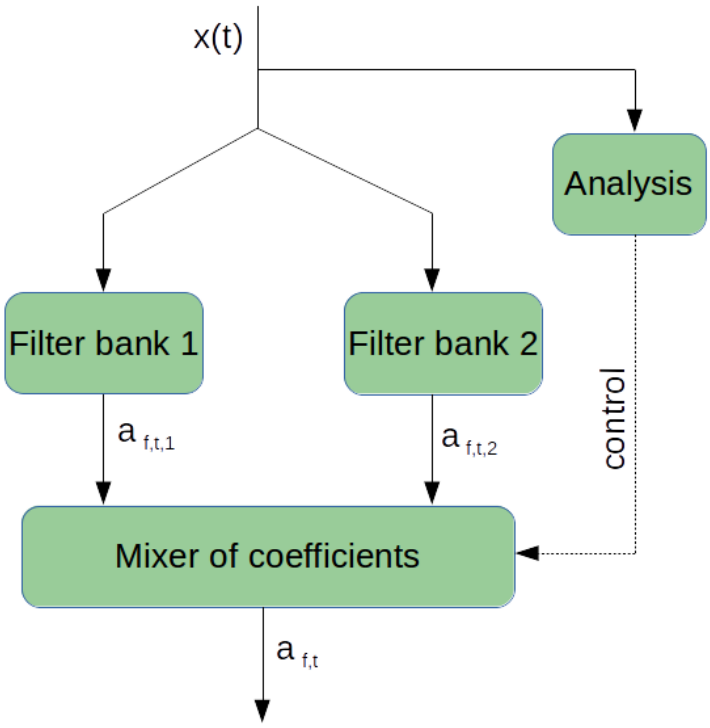


Figure 2.3: *A general structure for adaptive time-frequency representations (adapted from [LT06]).*

(STFTs with different window sizes, for example) and combined according to some type of information extracted from the signal. A specific algorithm is presented where different representations of a signal are compared according to their entropy: the TFP of a spectrogram is divided in small rectangular regions and for each subregion the representation with the smallest entropy is chosen, thus creating a multiresolution spectrogram. It is worth noting that this structure requires different representations of the entire signal to be computed before a choice is made on whether they are used or not for every subregion of the TFP, which means that a lot of data is computed and then discarded. Adaptive representations that follow this framework are prevalent in the literature and can be found in [dC20, LRM+13, JT07, CS10, JH17, KG13].

2.3 Sparsity measurements

An important conceptual part of adaptive representations is the quality of a given representation: given a set of representations of a specific region of the TFP, which one best represents the signal? As a rule of thumb, the sparsest representation is generally chosen to be the best. The sparsest representation is the one that maximizes the concentration or peakiness of information, being therefore the one that best localizes musical events (melodic notes or percussive hits) in the TFP.

Entropy is widely used for estimating signal information and complexity in the TFP. This is possible because entropy functions exploit the analogy between signal energy densities and probability densities [WBH91]: certain parallel characteristics between time-frequency representations and probability density functions suggest the Shannon entropy as a measurement for the complexity of a given time-frequency representation. Baraniuk et al. mention that "the negative values taken on by most TFRs prohibit the application of the Shannon entropy due to the logarithm" [BFJM01], but of course this can easily be corrected, since the range of the logarithm depend on the reference level (i.e. using the threshold of hearing or the minimum amplitude for a given bit-depth). This limitation can be avoided by the use of Rényi entropies, which form a class that generalizes the Shannon entropy. This class of functions is widely used in adaptive transform algorithms [LRM+13, BRSS19, JT07].

The Rényi entropy H_α of a discrete time-frequency representation $C[n,k]$ with time step δ_t and frequency step δ_f is given by:

$$H_\alpha(C[n,k]) := \frac{1}{1-\alpha}\log_2 \sum_n \sum_k \left(\frac{C[n,k]}{\sum_{n'} \sum_{k'} C[n',k']} \right)^\alpha + \log_2 \delta_t \delta_f.$$

As shown in [WBH91], the third-order Rényi entropy is immune to the negative time-frequency representation values and also measures signal complexity. A detailed review of its properties is given in [BFJM01]. Careful interpretation and analysis of the α parameter is needed, as different values of α determine different concepts of sparsity [LBR11]: as α increases, the difference between the entropy value of a sparse representation and a diffuse one increases. While high α values provide measures of entropy that are robust to noise, they are also less sensitive to weak spectral partials that should be taken into consideration in the measurement of musical contents. The parameter α, then, should be chosen taking into consideration this trade-off between robustness to noise and sensitivity to partials. An α value of 3 is commonly used [BRSS19, BFJM01], but other values can be useful: $\alpha = 0.7$ is chosen empirically in [LBR11], and with $\alpha = 0.3$ the entropy value is analogous to the power law that relates loudness levels in phons to perceived loudness in sones [LRM+13].

2.4 Sub-band processing

Sub-band processing is any form of processing that splits a signal into different frequency bands (whether by FFT analysis or a time-domain filter bank) and processes them separately. This type of processing is commonly used in audio compression algorithms and speech signal coders and transmitters, as well as in some multiresolution representations.

An efficient algorithm for the CQT based on sub-band processing is presented in [SK10]. In this implementation, the transform is performed an octave at a time, in the following manner: the implemented transform kernel produces only the coefficients for the highest octave of the audio signal. After these coefficients are calculated, the signal is low-pass filtered in order to remove the higher octave that has already been transformed. The signal is then subsampled by a factor of 2. This process is then iterated with the same transformation kernel, in order to obtain the coefficients for the next highest octave. This is repeated until all octaves of interest are processed. This algorithm takes advantage of the fact that the DFT frequency resolution is the ratio between the sampling frequency and the size of the analysis window. Since the signal is subsampled by two at each round and the analysis window (transformation kernel) remains the same, the resolution doubles for each subsequent octave, thus resulting in the desired constant Q factor. The computational complexity of this algorithm is that of the DFT times a fixed number of octaves desired.

Sub-band processing is also common in compression algorithms as part of the sub-band coding technique [TC79]: this type of algorithm begins with an M-channel filter bank. Each channel is then modulated in order to shift the center frequency to DC, low-passed and compressed (subsampled) by a factor according to the Nyquist frequency [CWF76]. This analysis process is depicted in Figure 2.4. The signal can be reconstructed by expanding each channel to the original sampling frequency by filling in with zeros, low-pass filtering the signal, re-modulating the signal back to the original center frequency and summing all channels.

A more efficient sub-band coding structure called integer-band sampling is presented in [VSW91]. After the filter bank analysis, the modulation, low-pass filtering and subsampling stages are substituted by a single subsampling step. Subsampling is performed in such a manner as to alias the signal in an advantageous way, in effect shifting the band-limited signal close to DC and achieving modulation and subsampling in one single step. This type of processing is also called **undersampling** [HP02, K+03] and is analyzed in further detail in Section 2.4.3.

Because of the computational efficiency of the above undersampling process, this appears to be a promising direction for developing a low-cost adaptive transform. This type of sub-band processing combines a computationally lightweight DFT analysis with a high frequency resolution for each band. For example, if a signal originally sampled at 40000 Hz is split into 20 different frequency

bands spanning 1000 Hz each, each band can potentially be subsampled (after modulation) at 2000 Hz. Then, if transformed with a DFT using a 500-sample analysis window, each band could be visualized with a frequency resolution of 4 Hz. In order to attain this resolution in the original signal with a regular DFT, it would be necessary to use a 10000-sample window. Each step of a sub-band processing algorithm for the proposed adaptive transform is detailed in the following subsections.

2.4.1 Band-pass filtering

Sub-band processing starts with a filter bank of time-domain band-pass filters (alternatively, a FFT filter bank can be used). Filter banks used in sub-band coding usually have 4 to 8 bands [CWF76, TC79] and each band-pass filter is designed to have sharp cutoff frequencies in order to isolate individual bands as much as possible.

Filter design takes into consideration the trade-off between smoothness and roll-off sharpness: ripples inside the passband and stopband allow for sharper roll-offs, but produce higher distortion in the signal [Ham98]. The choice of filter architecture is usually made between Butterworth filters (allow no ripples in the passband or stopband), Chebyshev type 1 filters (allow ripples in the passband), Chebyshev type 2 filters (allow ripples in the stopband) and elliptic filters (allow ripples in both the passband and stopband). Smooth filters are usually necessary for filter-bank applications that are concerned with the later reconstruction of the signal, since the overlap-add property is needed in the frequency domain.

Usually, every sub-band has the same bandwidth, but some adaptations can be made according to the application of the filter bank. For example, if trying to compress a piano recording to be used in an automatic music transcription algorithm, since the 8 kHz–20 kHz range does not contain fundamental frequencies of a typical 88-note piano, fewer bands may be used in that range as compared to the 20 Hz–8 kHz range.

2.4.2 Ring modulation

Ring modulation is a computationally cheap modulation technique in which a signal is multiplied by a simple modulator such as a sine wave. Take, for example, a signal $x(n)$ multiplied by $\cos(2\pi f)$; this operation results in a signal $y(n)$ that is a frequency-shifted version of $x(n)$. From the basic trigonometric property $\cos(a) \cdot \cos(b) = \frac{1}{2}\cos(a+b) + \frac{1}{2}\cos(a-b)$, it follows that each sinusoidal component of $x(n)$ will be shifted by $+f$ and $-f$ Hz.

This can be used to maximize the amount of decimation applied to a specific frequency band in a sub-band processing streamline. By modulating the band-limited signal close to DC and then low-pass filtering it, the resulting signal could be re-sampled with a frequency of twice the bandwidth and still accurately represent this band, as shown in Fig. 2.4.

2.4.3 Undersampling

Ring modulation can be substituted by a technique called undersampling, which is the sampling of a band-limited signal with a sampling rate below the Nyquist frequency, in order to create a low-frequency alias that is spectrally indistinguishable from the original signal [K+03]. According to the convolution theorem, the DFT of a sampled real-valued signal is composed of the spectrum of the signal and frequency-shifted copies of this spectrum called aliases. If these aliases don't overlap, it is possible to recover the original spectrum. Therefore, in order to obtain a useful low-frequency alias of the original band-limited signal by sampling it below the Nyquist frequency, it must be ensured that all spectrum aliases caused by this new sampling rate do not overlap. It is important to note that this condition must be met by both bands of a real signal (its positive spectrum and its conjugate-symmetric negative copy), since both are shifted by the sampling process.

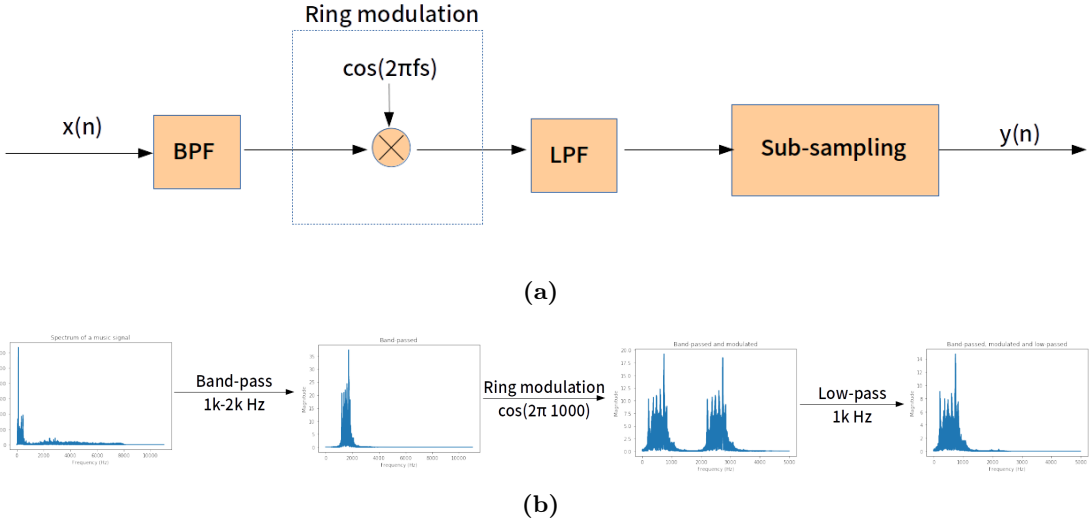


Figure 2.4: ***a)*** *A scheme for the compression of a single frequency band of a music signal* ***b)*** *The spectral view of the above process*

As shown in [Gas78], the conditions to avoid overlap between aliases of a band-limited signal between frequencies f_L and f_H are:

$$\frac{2f_H}{n} \leq f_s \leq \frac{2f_L}{n-1}, \tag{2.4}$$

where f_s is the new sampling rate and n is an integer satisfying

$$1 \leq n \leq \frac{f_H}{f_H - f_l}. \tag{2.5}$$

The highest n that satisfies both conditions leads to the lowest possible sampling rates. In Figure 2.5, the undersampling of a band-limited signal between f_l and f_h is depicted. The blue values represent the signal's original positive and negative spectra, the green values represent the aliased copies from the positive spectrum, the red values represent the aliased copies from the negative spectrum and the black values represent the useful low-frequency alias from the positive spectrum. Since there is no overlap between any of the aliases or original copies of the spectrum, undersampling was performed successfully and the black values are identical to the original positive spectrum of the signal.

2.5 Signal components and musical events

The term *component* is widely used in signal processing as a way to describe a concentration of energy in some domain, but there is no formal quantitative definition for it [Coh92]. In automatic music transcription, these components are usually called sound events, and can be characterized by their pitch, loudness, duration and timbre [KD07]. The usual definition is expanded here in order to include expressive elements, resulting in the following categories, here named **musical events**:

- Melodic events: notes with a clear fundamental frequency (as opposed to percussive events). The fundamental frequency and harmonic overtones of a given note constitute this type of musical event, and ideally should be frequency-located with precision. If a melodic note starts with a strong onset, this should be time-located with precision;

- Percussive events: notes such as cymbal and tom-tom hits, bells and other percussion instruments with no clear fundamental frequency. These events should be time-located with precision;

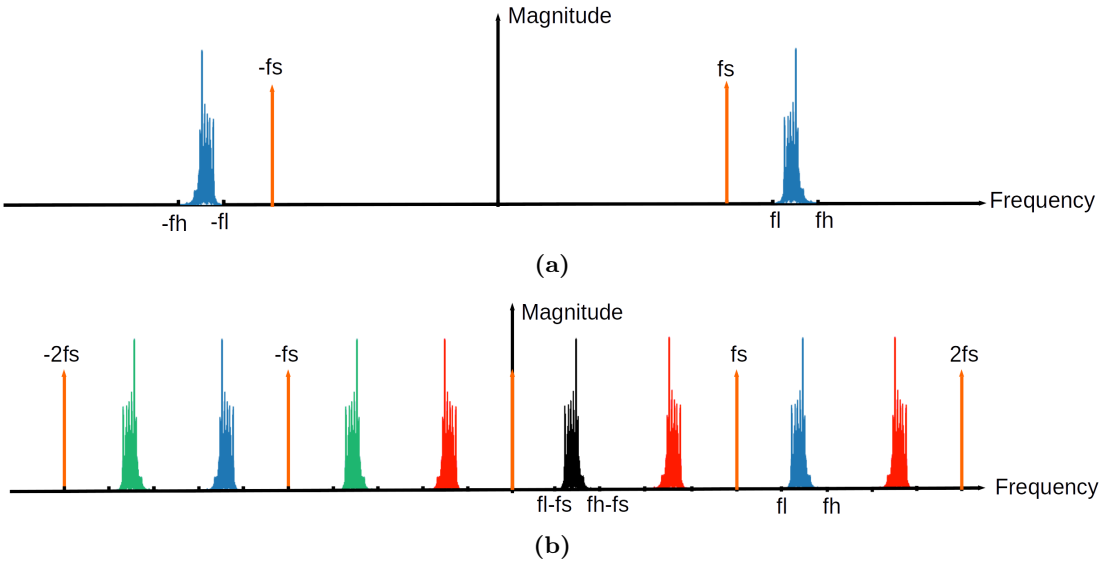


Figure 2.5: *A spectral view of undersampling:* ***a)*** *The spectrum of a signal band-limited between f_l and f_h and its new sampling frequency f_s.* ***b)*** *After subsampling the signal at f_s, several aliases of the original spectrum are produced. Since there is no overlap between any aliased copies, the black values represent a low-frequency alias that is identical to the original positive spectrum.*

- Expressive events: variations of the amplitude or frequency of a note, such as pitch bends, tremolo and vibrato.

These are the phenomena we wish to locate with time or frequency precision on a given time-frequency representation. In that sense, an ideal representation would be the one that best locates time and/or frequency-wise all musical events contained in a music signal. It is important to note that these phenomena are not sufficient to describe many musical pieces outside of the tradition of popular Western music, which produces a bias (also present in the utilized datasets) towards this style of music in the present work.

Chapter 3

Methodology

We start this chapter with an overview of our proposed adaptive multiresolution representation named the **Iteratively Refined Multiresolution Spectrogram (IRMS)**. We then present specific objectives to its design and the methodology proposed to reach them.

3.1 IRMS representation overview

As explained in Chapter 1, our motivations and discussion of related work lead us to the main objective of developing an efficient adaptive transform. This objective emerges as a counterpoint to the traditional adaptive analysis framework of the "jigsaw puzzle" [JT07], that combines several precomputed representations in different resolutions. Ideally, our representation would use very little detail to represent regions of the TFP that are not of interest (areas of silence, for example) and high detail in areas of the TFP that contain musical events and expressive elements. Instead of combining precomputed representations, our proposal is to achieve it with local refinements on top of an initial rough spectrogram, localized in the musically relevant subregions of the TFP. In other words, we would only pay the computational cost of a high resolution representation when justified, and not for the whole spectrogram. This method aims to save memory and computing power: it could use the same amount of data of a fixed-resolution spectrogram by using coarser resolutions in areas that do not contain musical events and higher resolutions in musically relevant regions, while at the same time reducing the computational overhead by avoiding throwing away computed data.

The proposed Iteratively Refined Multiresolution Spectrogram (IRMS) follows these steps (illustrated in Fig. 3.1):

1. An initial STFT representation is computed with a short window;
2. A music information estimator is extracted from this representation. This estimator indicates which subregions of the representation contain musical events (these are termed *relevant*), and thus should be represented with higher resolution;

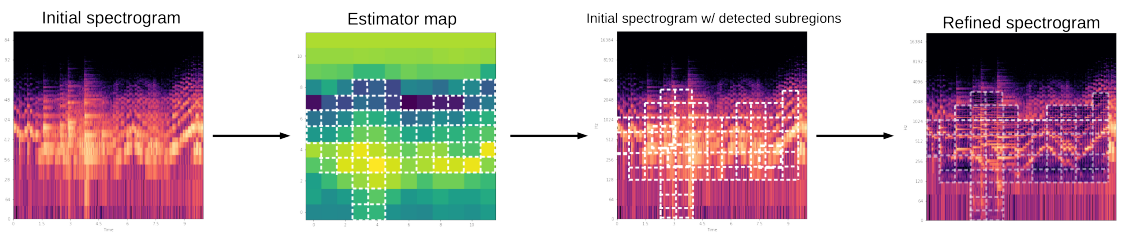


Figure 3.1: *A single refinement step of the IRMS representation. From an initial generic spectrogram, subregions (represented by the white dotted lines) are detected as musically relevant, and then transported to the original spectrogram in order to be refined via localized high resolution STFT computations. In the shown plots, brighter colors denote higher values (of either energy or musical relevancy).*

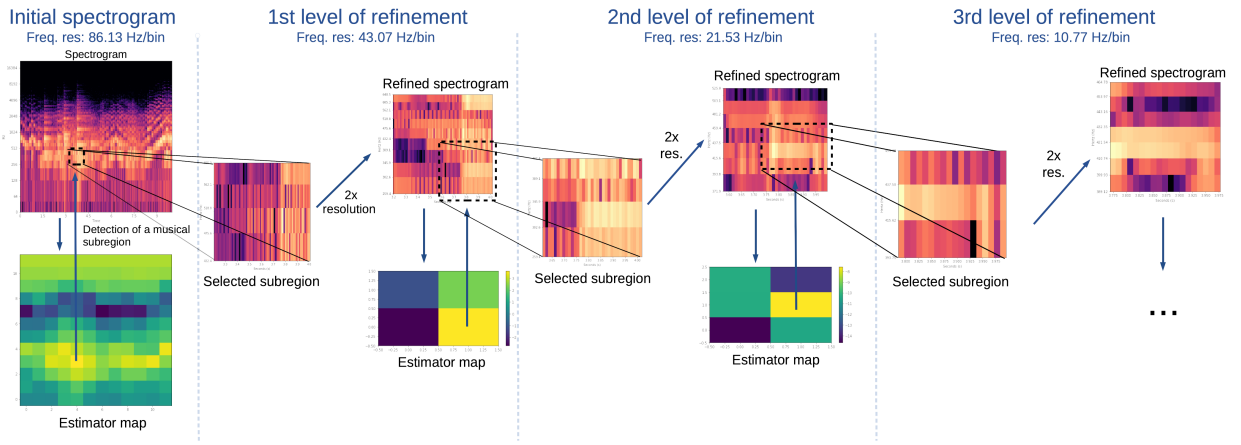


Figure 3.2: *The multilevel IRMS. We depict here the 3-level case, showing the refinement of only 1 subregion per level for the sake of clarity. On each level, an estimator map is computed, from which the most prominent subregion is picked and refined. The frequency resolution doubles at each level, and analysis subregion sizes (the size of each bin of the estimator map) get increasingly smaller with each level.*

3. Based on this estimator, some subregions are classified as musically relevant. This can be decided on a percentile or threshold basis;

4. Localized higher-resolution STFT representations of the subregions detected as relevant are computed using a sub-band processing algorithm;

5. These localized STFTs are inserted into the original spectrogram, resulting in a multiresolution spectrogram.

These steps define our algorithm with a single level of refinement. That is, detection of relevant subregions and subsequent refinement is performed only once. The algorithm can be repeated in a recursive fashion for several levels of refinement while decreasing the size of the analysis subregion, by expanding the original algorithm with these steps:

5. For every subregion refined in step 4, repeat steps 2-4. That is, inside every detected and refined subregion, determine which subsubregions are relevant and refine them further.

6. Repeat step 5 using the subsubregions refined by it in the previous step until a certain criterion is met (e.g. number of levels, execution time).

Fig. 3.2 illustrates the multilevel algorithm for a depth of three levels, where the frequency resolution is doubled at each level. This format enables a layered refinement, where increasingly smaller subregions are represented with an incresingly higher frequency resolution, thus producing a multiresolution spectrogram that prioritizes musically relevant subregions in a hierarchical fashion. As an alternative, we could also reach the same maximum resolution as in the presented scenario with a single level of refinement, by increasing 8 times the resolution of detected subregions. Although this could be computationally cheaper, it would create a binary (resolution-wise) spectrogram instead of a layered refinement.

The development of the IRMS depends on the investigation of two main problems: the identification of musically relevant regions and the efficient computation of representations of isolated rectangular regions of the TFP. In the following sections we break down each of these problems and explain the methodology used to investigate them.

3.2 Efficient STFT computation of isolated time-frequency regions

The task of computing STFT representations of isolated time-frequency regions can be achieved by sub-band processing (see Section 2.4). This type of processing has been extensively researched

and is common in signal processing algorithms, such as audio, image and video compression [Wal92, DLU91, LG91] and, more relevantly to the presented work, it is employed in the efficient CQT algorithm [SK10].

In order to explore these existing techniques and apply them to our task at hand, the development of an application called "STFT Zoom" is proposed.

3.2.1 "STFT Zoom" GUI application

A standard usage of the application should follow these simple steps:

1. The user selects an audio file to be inspected;
2. The application computes a low-resolution STFT of the audio file and displays it;
3. The user specifies a rectangular region of the spectrogram to be visualized in greater detail (optional resolution parameters can be specified);
4. A low-cost high-resolution STFT representation of this area is computed using sub-band processing and displayed to the user in a separate window;
5. If desired, other regions are specified and the process is repeated;
6. If desired, the high-resolution visualizations are overlayed on top of the original low-resolution spectrogram to create a multi-resolution spectrogram.

In addition to serving as an exploratory analysis tool, this application can be viewed as the first step towards a working algorithm for our proposed adaptive multiresolution representation, one in which the detection of relevant subregions is performed by the user. This is a way of detaching the sub-band processing problem from the task of automatic identification of relevant subregions.

Fig. 3.3 shows a visual prototype of the GUI of the application. Apart from its role in this research project, this tool could be useful as an application for manual sound analysis tasks, such as inspecting bird song recordings, analyzing human speech or helping in the manual transcription of music signals.

3.3 Identification of musically relevant regions of a spectrogram

The identification problem elicits the investigation of possible **music information estimators** to be applied on subregions of a time-frequency representation. In greater detail, we aim to answer the following questions:

- Is it possible to clearly identify the regions of a spectrogram that contain musically relevant information? What is the best estimator for this task? What is the necessary resolution of a spectrogram in order to perform this identification with satisfactory confidence?
- What are the appropriate dimensions of these subregions of the spectrogram to be analyzed? What are the trade-offs involved in this choice?

Two experiments are proposed in order to investigate these questions. First, we evaluate how well different estimators correlate to reference values relating to some MIR tasks. In the second experiment, we evaluate chosen estimators in a binary classification task to further evaluate their applicability as part of an algorithm for a multiresolution representation. The following subsections are heavily based on [FQ20], where they were first presented.

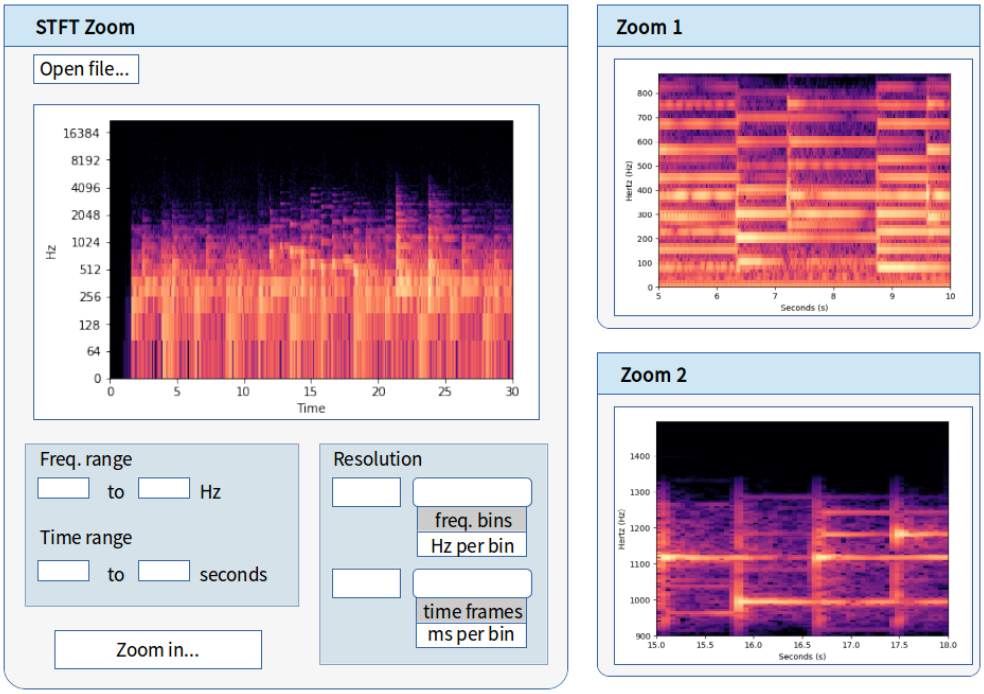


Figure 3.3: *Interface of the "STFT Zoom" application.*

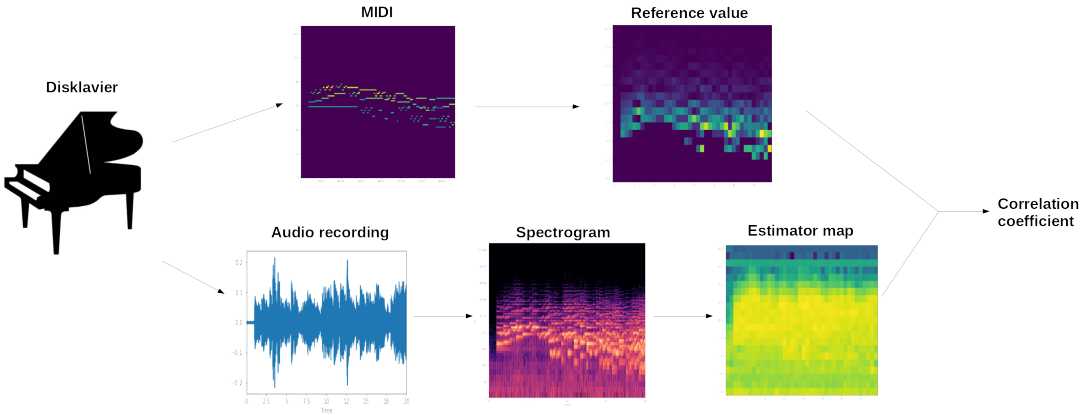


Figure 3.4: *Overview of the estimator evaluation experiment: from a single piano performance a MIDI-derived symbolic event density map and a spectrogram-derived estimator map are built and correlated. Reproduced from [FQ20].*

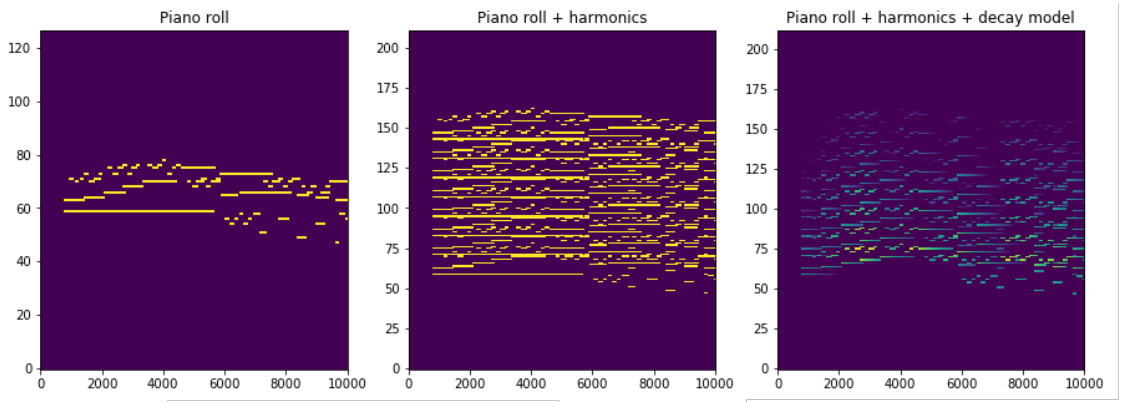


Figure 3.5: *Alternative intermediate symbolic representations for the event density map. Reproduced from [FQ20].*

3.3.1 Estimator evaluation

MIDI files are a representation of a song containing a description of each note and their duration. In a sense, they are sound event maps that can be used as ground truth in an experiment, since each of these notes can be precisely located in the TFP. The idea of this experiment is to compare a song's "true" sound event map with an estimated map computed from the spectrogram.

This is made possible through the use of piano performances recorded in a Yamaha Disklavier: for each performance in the utilized dataset, there are corresponding aligned MIDI and audio files. Fig. 3.4 gives an overview of the steps involved in this experiment: from the MIDI file, we build a density map of musical events (MIDI notes, possibly including harmonic partials of such notes), to be used as ground truth; from the sound recording, we build a density map using a candidate estimator. Both density maps are computed over rectangular TFP subregions measured in ms × cents (width × height). Finally, ground-truth and estimator density maps are compared according to their Pearson correlation coefficient, in order to see which estimator adheres best to the ground truth.

We now present the evaluated reference values and estimators.

Music information reference values

From each MIDI file, we extract three different reference values (see Fig. 3.5):

1. Piano roll: from the note-on/off and sustaining pedal position events, a simple binary piano roll representation (no velocity information) is built;

2. Piano roll with harmonics: the simple binary piano roll is augmented with the inclusion of 7 harmonics (also binary) for each note;

3. Piano roll with harmonics and decay models: a piano roll with velocity information is built from the MIDI events contained in the file, along with 7 harmonics for each note. Then, linear energy decay models are used to attenuate each note over the time axis and each harmonic over the frequency axis.

After their computation from the MIDI file, each matrix is divided into subregions with dimensions in cents × miliseconds. The mean value inside each of these subregions is considered in order to form the final reference matrix. It is important to note that the reference value (and thus our definition of music relevance) is strongly dependent on the subregion size. The above matrices represent the overall "note content" of each subregion, and so the size of subregions, along with the design decision of taking the mean value within each subregion, both directly influence the definition of music relevance.

Each estimator derived from the spectrogram was compared against each of these reference values, in order to give us insight into their applicability under different circumstances:

- The correlation of an estimator with the simple binary piano roll should give us an idea of its applicability to the AMT task, since a piano roll would be one possible format for the final product of a transcription. Thus, in a representation built specifically for AMT, piano roll events should be given more importance, and should be represented with a finer resolution with respect to the remaining subregions of the TFP.

- The correlation of an estimator with the piano roll with harmonics allows us to evaluate its sensibility to the presence of harmonics, which are an integral part of the timbre of nearly every musical instrument, and would show up on a spectrogram, but not necessarily in a music score. Harmonics might be of interest to timbre analysis, and so estimators that correlate well with this augmented piano roll would be candidates for producing multiresolution time-frequency representations for timbre-related tasks. Additionaly, there are F0 detection algorithms that utilize harmonics in their computation [SGER14], so this ground-truth value is also related to AMT.

- Finally, the introduction of decay models into the piano roll represents an attempt to include a very simple acoustic instrument model: this may be suited to test the invariability of each estimator in relation to specific instrument timbres, but also to look for estimators that would be useful to study dynamic (i.e. time-varying) aspects of timbre.

Estimators

Rényi entropies [LRM+13, BRSS19, JT07] and energy variance [ZN77] have both been employed as musical information estimators in the TFP. The Shannon entropy was also considered in our evaluation as an alternative information estimator. The standard deviation was used in place of variance, in order to preserve the same scale of the original (energy or amplitude) data. All of these estimators were extracted from amplitude, energy and dB energy STFT spectrograms. Finally, the amplitude, energy and dB energy densities (i.e., their mean values inside each subregion) were evaluated, totaling 12 estimators.

These estimators are motivated by the fact that musical events are characterized not only by an increase of energy or amplitude (which would be captured by their densities) but also by an increase of information complexity due to note onsets, frequency gaps between harmonics and other observable events. This increase in complexity (possibly captured by entropy and standard deviation) also justifies a closer look at these regions: an adaptive transform should use a finer resolution to represent regions containing onsets, for example, but also regions containing entangled harmonics belonging to different harmonic series (i.e., different notes).

3.3.2 Binary classification of TFP subregions

In the second experiment, the best performing estimators of the first experiment are to be further evaluated as features to be used in a predictive model for the binary detection of musically relevant regions of the TFP. This experiment follows the same structure of the first one: ground truth values are extracted from a MIDI file and features are extracted from the corresponding audio recording. Single-feature predictive models and feature pairings will be tested in the training of a Gaussian Naive Bayes model.

Firstly, the binary piano roll will be used as ground truth. After its decomposition into subregions, each subregion that contains at least one note will be labeled as relevant. As discussed in Section 3.3.1, a feature that accurately predicts subregions containing notes would be a good detector to be used as part of an AMT-motivated multiresolution representation. Secondly, in order to evaluate the influence of harmonics in the detection, a ground truth consisting of a binary representation of the piano roll with harmonics and decay models will be used. For this reference

value, a musical density threshold has to be chosen in order to use this map to produce binary (relevance) labels (further discussion is presented in Sec. 5.2). The comparison of these models with the previous ones should give us insight into the sensitivity of different estimators to harmonics, and how these harmonics aid or harm the detection of musically relevant TFP regions.

As a result of these two experiments, we will have an evaluation of the best performing estimators for the detection of musically relevant subregions of a TFP representation, along with a trained Naive-Bayes model for this task.

3.4 Integration and final evaluation

The final stage in the development of the IRMS is the integration of the automatic identification of musically relevant regions of the TFP with the sub-band STFT processing of such regions. This stage defines the translation between an estimated map of musical event density and the computation of a multiresolution spectrogram. This integration is dependent on the following choices:

- The mapping between a subregion's estimated density of musical events and whether it is considered relevant (i.e. a subregion that should be represented with higher resolution) or not. This could be done by a percentile choice (always refine 50% of the spectrogram, for example) or a threshold choice informed by the experiments shown in 3.3. Plus, there is the choice of STFT resolution used to represent each detected subregion. Should this be fixed for each level of refinement or follow the values of the estimators (i.e., should musically denser subregions be represented with higher resolution)?
- The data structure used to represent the multiresolution spectrogram. What would be an efficient structure, i.e. one that is sparing in memory usage and which facilitates search and insertion of data? How to efficiently produce a visualization of such multiresolution spectrogram?

These are more so guiding choices in a specific implementation of the IRMS than guidelines for an exploratory experiment. These do not concern the conceptual ideas behind the IRMS, but do play a factor in some important characteristics of it, such as its execution time and memory used. The following experiment aims to evaluate the integrated algorithm and to supply data in order to inform the choices above.

3.4.1 Computational cost experiment

Given that computational cost is one of the main motivations behind this work, it should be one of the evaluating factors of the final solution as well as a strong guiding factor in the choices related to algorithm integration. In this experiment, we intend to profile several configurations of the IRMS and to compare them with different existing representations for music signals based on execution time.

The IRMS representation is here compared to the following time-frequency representations:

- **STFT**: the fixed-resolution representation used by the IRMS as its foundation;
- **Constant-Q Transform (CQT)** [SK10]: a multiresolution non-adaptive representation with fixed resolution per octave;
- **Sample-Weighted Geometric Mean (SWGM)**: proposed in [MdVB17], it is a bin-wise combination of a previously computed dictionary of representations, whose main idea is to "combine the spectrograms in such a way that the lowest valued sample takes precedence, increasing the sparsity [of the resulting multiresolution spectrogram]" [dC20]. Since this combination takes into account the values presented by the spectrograms, it can be considered an adaptive multiresolution transform;

- **Lukin & Todd's maximal energy compaction principle** [LT06]: an adaptive representation that combines a previously computed dictionary of representations based on a local sparsity criterion (measured by an entropy-like feature) that intends to minimize energy smearing in both time and frequency axes;

- **Smoothed Local Sparsity (SLS)** [dC20]: a representation analogous to that of Lukin & Todd, that also combines previously computed representations based on a sparsity criterion: it performs a mean of samples weighted according to their local sparsities (measured by the Gini index), while ensuring soft transitions between different resolutions.

Although the competing techniques are designed based on different goals, in this experiment we adopt a one-criterion-fits-all, namely the requirement that each representation attains a given target frequency resolution in its most refined regions of the TFP. For each multiresolution representation, it is left to each algorithm to decide, based on its intrinsic models, exactly which subregions of the plane should attain the target frequency resolution. The STFT spectrogram will present this target resolution homogeneously, while multiresolution representations will present this resolution only on some of its bins. In the case of methods based on the combination of previously computed representations (SWGM, Lukin & Todd, SLS), this maximum resolution is equal to the resolution of the finest (frequency-wise) representation in the provided dictionary.

The computation of these representations will be timed for several target frequency resolutions spanning from 86.13 Hz/bin to 1.35 Hz/bin (the equivalent of using 512 and 32768 sample windows in a 44100 Hz signal, respectively) across a dataset of music signals. We take the mean of all files for each resolution point, and, as a result, we will have a curve of execution time as a function of the maximum frequency resolution, for each one of the tested representations. With this experiment, we aim to answer the following questions:

- With regards to the IRMS, what is the influence of the analysis subregion size in its cost? What is the influence of the number of refinement levels in its cost? How does its cost vary from file to file?

- How does our solution compare to the others? How efficient is it compared to other adaptive multiresolution representations? Does it behave similarly in regards to cost versus maximum resolution (i.e. linearly, exponentially)?

Chapter 4

Implementations

In the first section of this chapter, we give technical details and observations about the "STFT Zoom" tool. It functions as an interface for computing STFT representations of localized subregions of the TFP, using a sub-band processing algorithm. This tool is a first step towards our proposed IRMS representation, that uses the same sub-band algorithm along with automatic detection of relevant subregions to create a multiresolution spectrogram. Implementation details and integration decisions for the final algorithm of the IRMS are presented in the second section of the chapter, along with its complexity analysis.

4.1 Sub-band processing algorithm for "STFT Zoom"

The sub-band processing algorithm implemented in the "STFT Zoom" application follows these steps (see Figure 4.1):

1. A frequency range and time range are specified by the user;
2. Optionally, the user specifies the desired frequency and time resolution of the zoomed-in region. The frequency resolution can be specified in Hz/bin or number of bins, and the time resolution can be given in ms/frame or number of time frames;
3. The audio file is sliced in the specified time range;
4. Depending on the lower limit of the frequency range, the sliced audio file is either band-passed or low-passed in order to isolate the specified frequency band (see 4.1.1);
5. If band-passed in the previous step, the algorithm tries to find an undersampling frequency (see 4.1.2). If found, the audio is undersampled, in order to perform modulation and subsampling in one single step. If not, the signal is modulated and low-passed once more in order to get rid of the alias caused by ring modulation (see 4.1.3).
6. If not undersampled in the previous step, the signal is subsampled using the appropriate frequency described in 4.1.3;
7. An STFT is computed for the signal obtained in the previous steps. The analysis window and hop size are chosen according to the resolution specified by the user.

The filtering, undersampling, modulation and subsampling stages are detailed in the next subsections.

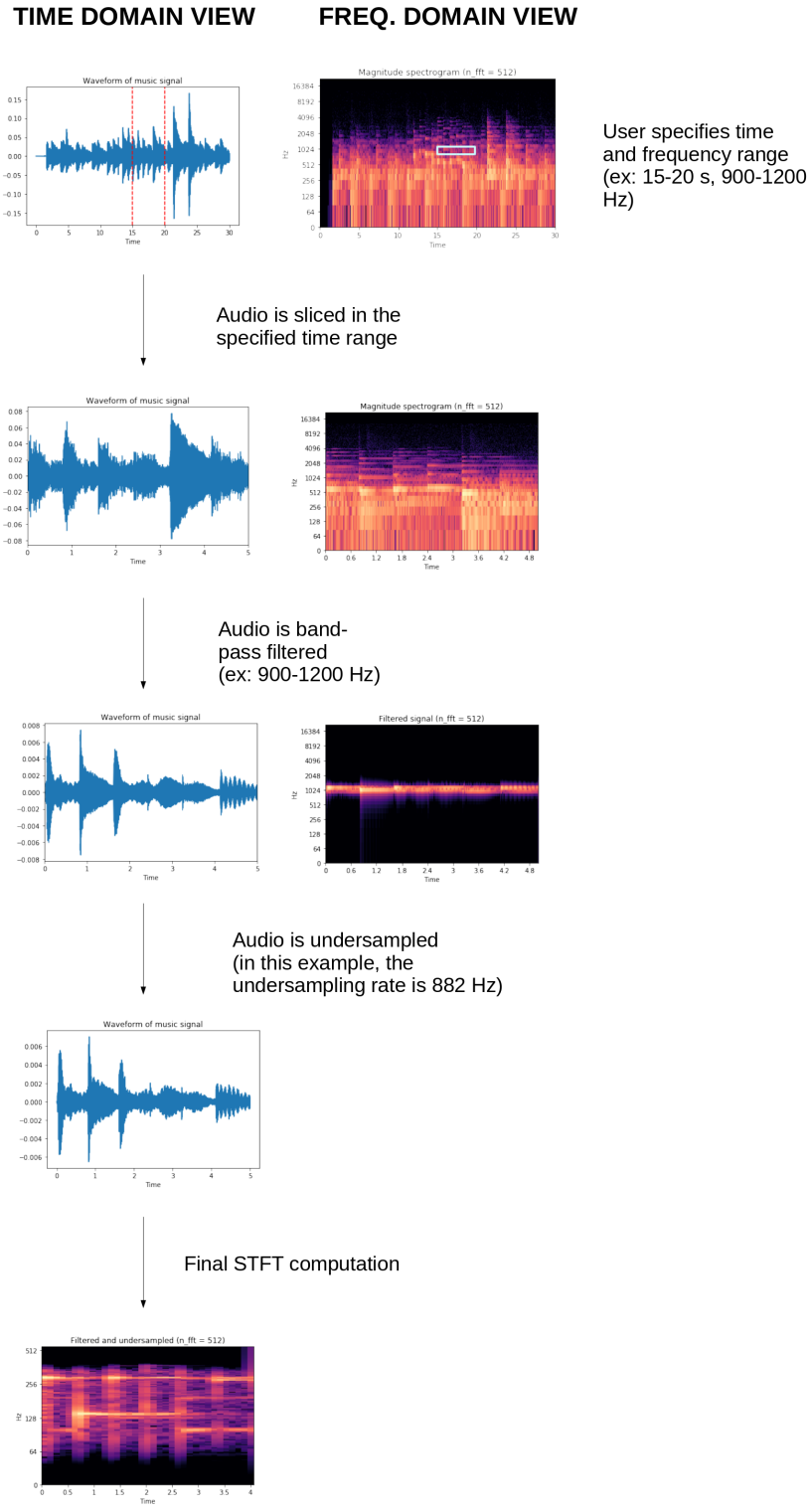


Figure 4.1: *A time domain and frequency domain view of the sub-band processing algorithm implemented. It is important to note that all computations (slicing, filtering, undersampling) are performed in the time domain signal, and the frequency domain view is only included for illustration.*

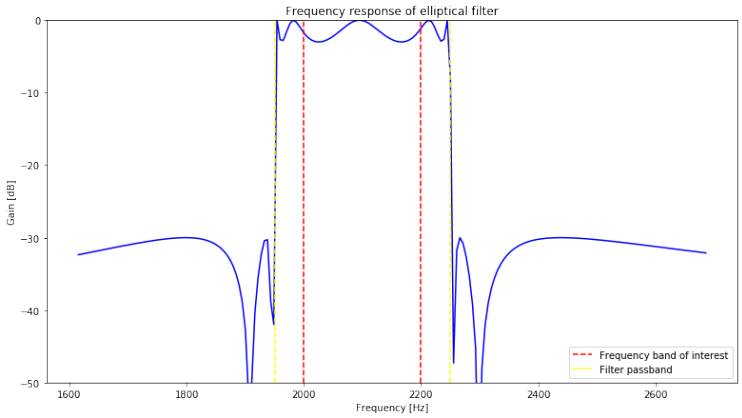


Figure 4.2: *Amplitude response of an elliptical bandpass filter with passband frequencies equal to 1950 and 2250 Hz (shown in dotted yellow lines). This figure depicts the filter corresponding to the 2000–2200 Hz frequency band (shown in red dotted lines).*

4.1.1 Band-pass filter and low-pass filter

As discussed in 2.4.1, there is a trade-off between smoothness of a filter's frequency response and the steepness of its roll-off when choosing its architecture. For the present application, a steeper roll-off means that lower rates of subsampling can be used after the filtering stage, which is very advantageous for an efficient high resolution STFT computation afterwards. On the other hand, ripples in the filter's passbands (that make a steeper roll-off possible) could distort the signal's characteristics inside the frequency band of interest, which is not desirable for an application primarily concerned with the representation of signals.

With this trade-off in mind, the Elliptical filter architecture was chosen for band-pass filtering. The filter is designed to have no more than 3 dB of attenuation in the passbands and at least 30 dB of attenuation in the stopbands, with a maximun filter order of 7. Passbands are chosen as the frequency range specified by the user extended by 50 Hz in each edge (see Fig. 4.2). This way, there are safeguards preventing the distortion of the frequency band of interest: the (at most) 3 dB distortion allowed inside the band is negligible. These parameters were chosen in order to ensure small filter orders (that speed up computation), steep roll-offs and smoothness inside the frequency band of interest.

If the specified frequency range has a lower limit of 400 Hz or less, the band-pass filter is substituted by a low-pass filter. Again, an elliptical filter is used, designed with a 3 dB ripple allowed in the passband and a minimum attenuation of 30 dB in the stopband. The filter's critical point, which is the point where the filter gain first drops below 3dB, is chosen as the upper limit of the frequency range specified by the user. A maximum filter order of 7 is permitted (see Figure 4.3). This same filter is used for the case where an undersampling frequency cannot be found in step 5 above, in which case the signal has to be ring-modulated and low-pass filtered.

4.1.2 Undersampling

As described in Section 2.4.3, modulation and subsampling can be achieved in one stage using a technique called undersampling. In order to determine a new sampling rate f_{under} that successfully performs undersampling, the following steps are taken:

1. The highest n that satisfies Equation 2.5 is determined, using the pass-band filter stopbands as f_L and f_H;

2. This n is used to determine the bounds of f_s in Equation 2.4;

3. The subsampling rates that are achievable without interpolation (by selecting 1 sample out of each M original samples) and that lie between these bounds are listed. If there are multiple

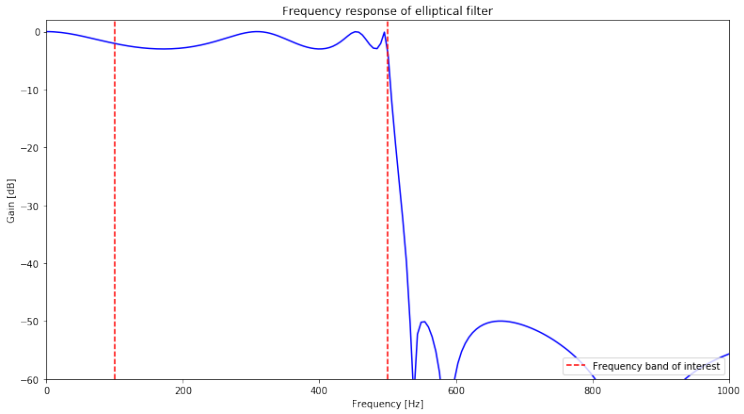


Figure 4.3: *Amplitude response of an elliptical lowpass filter with cutoff frequency set at 500 Hz. This would be the filter used to isolate the 100–500 Hz frequency band (shown in red dotted lines).*

rates, the highest one is returned;

4. If a subsampling rate cannot be found in step 3 and $n > 2$, the algorithm returns to step 2 with $n = n - 1$. If $n = 2$, undersampling cannot be performed in this band-limited signal.

Undersampling will shift the frequency band of interest $[f_L,\ f_H]$ in the following manner:

- If the undersampling frequency f_{under} was found using an odd n, the frequency band $[f_L,\ f_H]$ will be shifted by $-\lfloor f_L/f_{under} \rfloor \cdot f_{under}$ Hz;
- If the undersampling frequency f_{under} was found using an even n, the frequency band $[f_L,\ f_H]$ will be mirrored (the alias closest to DC is an alias of the symmetric negative spectrum) and will be shifted to the interval $[-f_H + \lceil f_H/f_{under} \rceil \cdot f_{under},\ -f_L + \lceil f_H/f_{under} \rceil \cdot f_{under}]$. It is important to note that in this case, after the STFT computation of the undersampled signal, the spectrogram must be inverted in the $[-f_H + \lceil f_H/f_{under} \rceil \cdot f_{under},\ -f_L + \lceil f_H/f_{under} \rceil \cdot f_{under}]$ frequency range in order to represent the positive spectrum of the signal.

4.1.3 Modulation and subsampling

If no undersampling frequency is found, ring modulation is performed. The signal is shifted in $f_L - 150$ Hz (a safeguard of 150 Hz is used in the band-pass filter), bringing the frequency band of interest close to DC. Since ring modulation creates an alias shifted on the other direction, the signal is low-passed after modulation (see Fig. 2.4.2).

Finally, subsampling is performed by choosing the closest frequency f_{sub} that satisfies the following conditions:

- $f_{sub} > 2 \cdot (f_c + 100)$, where f_c is the cutoff frequency of the low-pass filter; this is the Shannon-Nyquist condition with a safeguard of 100 Hz;
- $f_{sub} = f_s/i$, where i is an integer and f_s is the original sampling rate; this way, subsampling can be performed without interpolation.

4.1.4 Observations and computational details

Figure 4.4 shows the implemented "STFT Zoom" GUI application, which is a working tool for the detailed visualization of specific regions of a spectrogram, implemented as described in this chapter. The application is written in Python, and its structure is quite simple: a graphical interface built with the TkInter package gets the necessary parameters from the user and calls a script that performs the appropriate sub-band processing algorithm. The STFT result is then plotted using

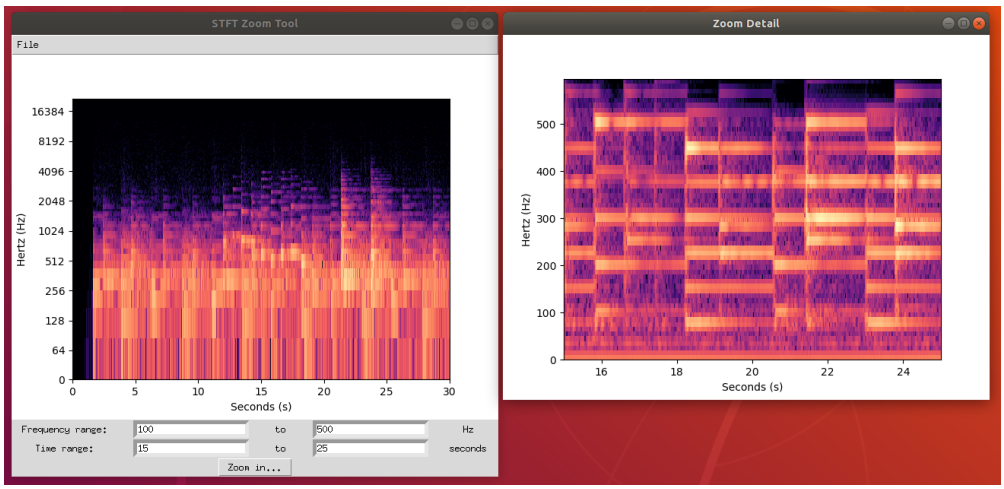


Figure 4.4: *A screenshot of the implemented "STFT Zoom" application, showing an initial spectrogram of a music signal and a detailed view of the 15 to 25 seconds and 0 to 500 Hz sub-region.*

Matplotlib inside a new TkInter window. SciPy is used in the design and application of both filters, LibROSA [MMB+19] is used for STFT computations and conversion between amplitude and dB spectrograms, and NumPy is used throughout for representation and computations involving arrays and matrices. The code is available at https://github.com/nicolasfigueiredo/stft-zoom.

4.2 Iteratively Refined Multiresolution Spectrogram (IRMS)

In this section, we run through each stage of the IRMS discussing implementation details and decisions made on the integration between the detection of musically relevant subregions and their refinement. We also take a look at the chosen data structure to store the IRMS and the code organization, along with an analysis of its computational complexity and example plots of the final working IRMS representation. An important observation is that some of the implementation details, such as choice of features, are informed by the experiments in Cap. 5. These parts of the algorithm are presented generically (as in "plug your feature here"), and specifics are left as a user decision.

4.2.1 The algorithm and integration decisions

The IRMS algorithm starts off with the computation of the initial rough STFT. This is done with default values of 512 samples for the window and hop size, although different values can be chosen by the user. Then, the IRMS data structures are initialized (more details on 4.2.2) with this spectrogram as its base, and the first round of detection of musically relevant regions is performed.

Subregion detection

Detection of relevant subregions depends on the following parameters supplied by the user:

- Subregion size: determines the size of the subregion (in ms per cents) from which a music information estimator will be extracted;
- A predictive model: this model takes as input the music information estimator and outputs, for each subregion, a probability that the subregion contains a note;
- A percentile or threshold choice: this determines the subregions that will be passed on to the refinement stage. If a percentile p is given, this means that the top $p\%$ of subregions (according to the probability computed by the model) will be refined. If a threshold t is given, this means that subregions with probability above $t\%$ will be refined.

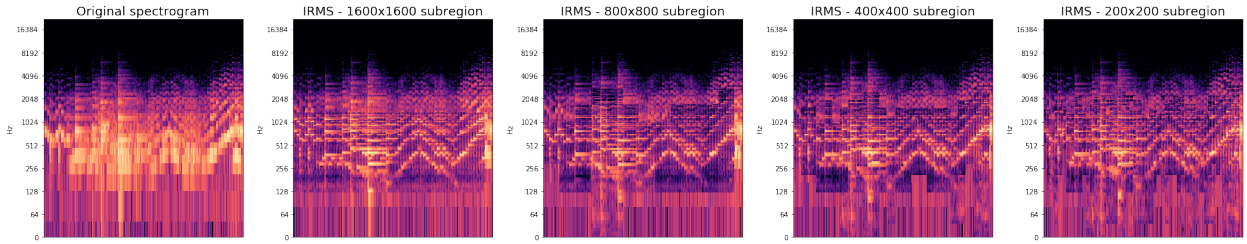


Figure 4.5: *IRMS visualizations showing the influence of subregion size used in the detection stage. A refinement percentage of 35% and k factor of 8 were used for all plots. Although a smaller subregion size produces a more selective detection process, it also results in the sub-band processing algorithm being called more times, which could mean a longer execution time for the same refinement percentage.*

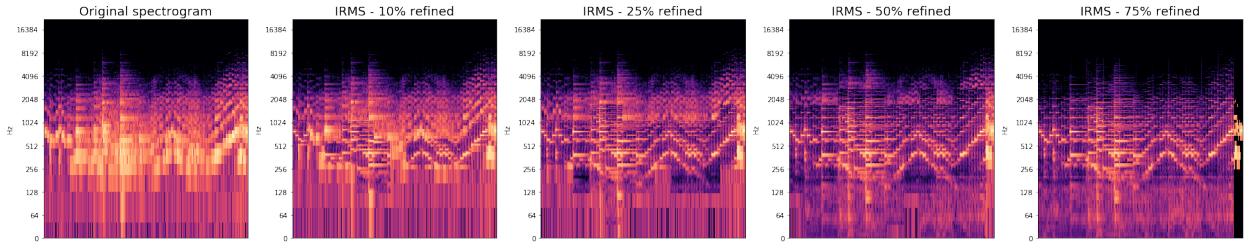


Figure 4.6: *IRMS visualizations showing the influence of refinement percentage used in the detection stage. A subregion size of 800 ms by 800 cents and k factor of 8 were used for all plots.*

Given these parameters, the first step in this phase is feature extraction from the initial spectrogram. According to a subregion size given in cents × ms, the spectrogram matrix is split into rectangular regions as follows:

1. The array of frequencies `fft_frequencies` that define the y-axis of the spectrogram is computed. Then, certain frequencies of this array are selected in order to create a new array `subregion_freqs` that presents the following property: the musical interval defined by two consecutive elements of this array is approximately the subregion height given in cents. This approximation is a result of the fact that `subregion_freqs` is a subset from `fft_frequencies`, in order to avoid splitting a single spectrogram bin across two subregions. This is done with Algorithm 1, using function `find_nearest(value, array)` that returns the element of `array` that is nearest to `value`.

2. The column step is determined as the closest integer to the ratio between the subregion width $t_{subregion}$ and the time width of a frame of the spectrogram t_{frame}.

3. Given this column step and the frequencies that define the borders of each subregion, a double-nested loop slices the spectrogram, selecting the bins pertaining to a subregion, and calculates the music information estimator. The result of this double-nested loop is a feature matrix to be passed on to a predictive model.

With this feature matrix as input, the model calculates a probability of music relevance for each subregion. Then, according to a percentile or threshold supplied by the user, some of the subregions are selected to be refined. The detected subregion indices are translated into the corresponding time and frequency ranges (e.g. subregion 1 is the area of the TFP defined by intervals 1-2 seconds, 200-400 Hz), finalizing the detection portion of the algorithm. In the translation between subregion index and time/frequency range, we pay attention to the fact that both the x-axis and y-axis values of the spectrogram are represented by center values, and not the border values of each bin. We need to include safeguards of half the time step and half the frequency step in the respective ranges in order to properly perform this translation.

Figs. 4.5 and 4.6 show the influence of subregion size and refinement percentage in the IRMS for a 10 second excerpt from a piano recording taken from the MAESTRO dataset. As illustrated

Algorithm 1 Function find_freq_list(fft_frequencies, delta_cents):

$f_1 \leftarrow 20.0$
$f_2 \leftarrow f_1 \cdot 2^{delta\_cents/1200}$
$idx_{f_1} \leftarrow$ find_nearest(f_1, fft_frequencies)
$idx_{f_2} \leftarrow$ find_nearest(f_2, fft_frequencies)
idx_list is initialized with $[idx_{f_1}]$
if idx_{f_1} equal to idx_{f_2} **then**
  $idx_{f_2} \leftarrow idx_{f_2} + 1$
end if
while $idx_{f_2} <$ length of fft_frequencies **do**
  Append idx_{f_2} to idx_list
  $idx_{f_1} \leftarrow idx_{f_2}$
  $f_1 \leftarrow$ fft_frequencies$[idx_{f_1}]$
  $f_2 \leftarrow f_1 \cdot 2^{delta\_cents/1200}$
  $idx_{f_2} \leftarrow$ find_nearest(f_2, fft_frequencies)
  if idx_{f_1} equal to idx_{f_2} **then**
    $idx_{f_2} += 1$
  end if
end while
Append the last element of fft_frequencies to idx_list
return idx_list // these are the indices of fft_frequencies that define subregion_freqs

in Fig. 4.5, using a smaller subregion size produces a more selective detection process, and, as a result, it allows for the refinement of the whole signal while selecting a smaller percentage of the TFP (this observation is reaffirmed by the results in Sec. 5.4). However, a smaller subregion also results in the sub-band processing algorithm being called more times and on top of increasingly smaller frequency bands, which could result in a longer execution time. Fig. 4.6 shows that, for this example recording, the IRMS algorithm behaves as desired, first refining the area between 100 and 1000 Hz that contains the majority of music information, and then expanding to subregions containing harmonics. It also indicates how a refinement percentage above 50% could be excessive for representing the music information of most recordings.

Before we move on to the refinement stage, we have to discuss a proposed algorithm optimization: the computation of a signal bank.

Signal bank

During development and evaluation of the IRMS algorithm, it became clear that its most time-consuming parts are the filtering stages (see Appendix 6.2). It became also clear that most detected relevant subregions were located between 100 and 1500 Hz. With this in mind, a code optimization was proposed, in which a bank of pre-filtered and modulated signals is computed as follows (see Fig. 4.7):

1. In the same way as step 1 of the subregion detection, the frequencies that define the subregion bandwidths are computed.
2. For each defined frequency band between 100 and 1500 Hz, we apply a band-pass filter on the whole signal, modulate the result shifting it close to 0 Hz, and then apply a low-pass filter (more details on this sub-band processing algorithm are given in Sec. 4.1.
3. We then store these filtered and modulated signals in an array, to be consulted later on in the refinement stage.

If a relevant subregion is detected between 100 and 1500 Hz, we then only need to select the corresponding pre-filtered signal from the bank, slice it, subsample it and perform the final STFT.

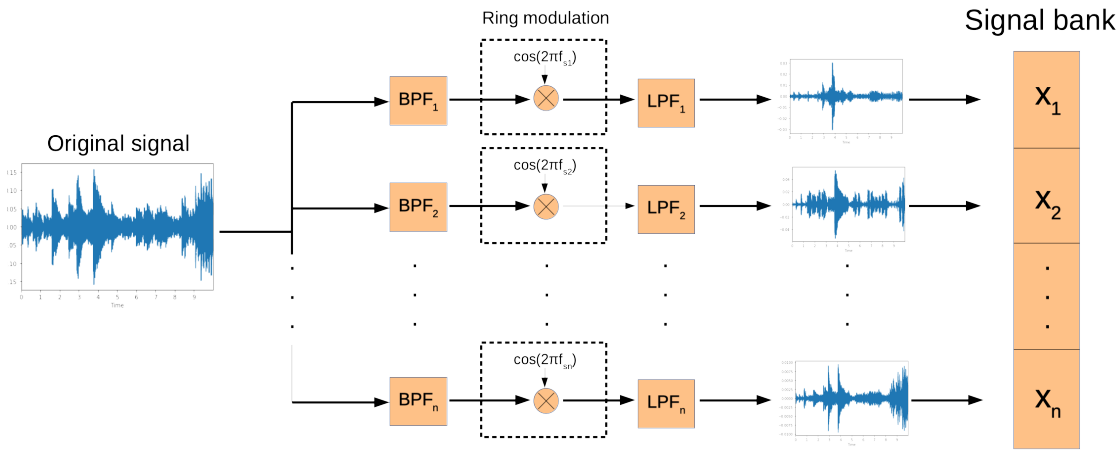


Figure 4.7: *A signal bank for faster computation of the localized STFTs. The frequency band between 100 and 1500 Hz is divided into musical intervals equal to the subregion size in cents, and each band-pass filter (illustrated as BPF) isolates each interval. Then, each interval is modulated close to DC and stored into an array. This array is consulted in the sub-band processing part of the IRMS.*

Without this optimization, we would have to perform band-pass filtering and low-pass filtering for each subregion detected between 100 and 1500 Hz, instead of only once per frequency band.

This optimization was empirically evaluated on the same dataset used in the experiments in Chapter 5, using 30 second excerpts from the piano recordings. The signal bank significantly improved the execution time of the IRMS by orders of magnitude. Nonetheless, since this improvement is highly dependent on recording duration and musical content of the tested recordings, it should be re-evaluated when used in different datasets.

Refinement stage

The detection stage of the algorithm produces a list of frequency and time intervals that define the subregions that were detected as musically relevant. For each element in this list, we perform the following computations:

1. Check if the subregion lies between 100 and 1500 Hz. If so, select the corresponding signal from the signal bank, slice it in the subregion's time range and go to step 3;
2. If not, perform the sub-band processing algorithm as defined in Sec. 4.1 in order to filter and modulate the signal;
3. Subsample the signal according to the minimum sampling rate as discussed in Sec. 4.1;
4. Compute the STFT of this subsampled signal, according to a desired frequency and time resolution based on the k factor (to be discussed in 4.2.1). This is the refined representation of the detected subregion.
5. Insert this refined subregion into the IRMS data structure that stores the multiresolution spectrogram. If desired, refined subregions may be normalized before insertion (see 4.2.2 for details on how this is done).

STFT computations usually center the first analysis window at the signal's first sample and use zero padding in its beginning and end for calculation of the first and last time frames. In order to substitute this padding with actual information from the signal, when performing a localized STFT we extend the time range analyzed in half the window size in both directions. By doing this, disabling padding and ensuring that the first window starts at (instead of being centered at) the first sample of the analyzed signal, the first and last windows will be centered at the beginning and

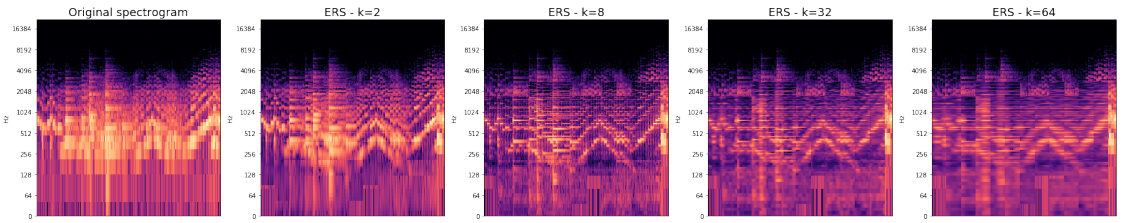


Figure 4.8: *IRMS visualizations showing the influence of the k factor in the final representation. A subregion size of 800 ms by 800 cents and a refinement percentage of 50% were used for all plots. From left to right, the frequency resolution of the refined subregions are the same as using window sizes of 1024, 4096, 16384 and 32768, respectively, for a 44100 Hz sampling rate signal.*

end of the original time range, respectively. This does not expand the time range of the resulting STFT beyond the original time range of interest, it only substitutes padding for information present in the original signal.

In practice, the sub-band processing algorithm is the same as the one implemented in the "STFT Zoom" tool (Sec. 4.1), but with the aid of the signal bank and a simple change to the filtering stages. In order to ensure quicker execution times for the IRMS, the pass-band is extended in 250-Hz in both directions when performing band-pass filtering, and in 100-Hz when performing low-pass filtering. Since we keep the filter design requisites (stop-band attenuation, pass-band ripple) fixed, a larger pass-band means a smaller filter order, which is computationally cheaper. After the computation of the localized STFT, the resulting matrix is sliced in the original frequency range of interest. The resolution of the refined subregions is defined by the k factor, discussed next.

The k factor

Every level of refinement has its frequency resolution defined by a k parameter supplied by the user. This means that resolution is uniform per level, and is defined in relation to the resolution f_{res}^{base} of the initial base spectrogram. In simple terms, frequency resolution f_{res} of level l is defined as $f_{res}^{l} = f_{res}^{base}/k_l$ (since resolution is given in Hz/bin, a smaller value configures a finer representation). The window size of each localized STFT is defined according to this resolution, taking into account the subsampling rate achieved by the sub-band algorithm.

Hop size is calculated in order to keep time resolution fixed for all subregions and all levels of refimenent, and equal to the time resolution of the initial base spectrogram. This means that when refining a subregion, we keep the same number of time frames as in the base spectrogram. It is worth noting that, while the number of time frames is fixed, the temporal region covered by each STFT bin increases with the k factor, since we are increasing the size of the analysis window used to compute each time frame. This increased overlap should be taken into account when extracting temporal information (e.g. onset times) from the multiresolution spectrogram. In other words, there is no running away from the uncertainty principle. We consider as future work the development of a more complex detection algorithm, capable of detecting not only relevant subregions of the TFP, but whether they should be refined time or frequency-wise. In the current implementation, the IRMS is also capable of refining subregions time-wise instead of frequency-wise, by setting the k parameter with a value lesser than 1.

Fig. 4.8 shows the influence of the k factor in the final IRMS visualization. As discussed, very high values of k result in spectral smearing in the direction of the time axis. The visualization of high k values is also limited by the issue described later in 4.2.2: the actual resolution of the IRMS visualization is bounded by the size chosen for the final image, which is kept constant for this example.

Normalization

Normalization is a necessary step in order to match the energy of spectrograms of a given signal computed with different window sizes, and, in the case of the IRMS, versions of a signal with different sampling rates, modulated and filtered in different ways. If normalization is not performed, energy values across the multiresolution spectrogram will not be comparable across subregions, and a subsequent AMT algorithm that accesses the IRMS like a fixed-resolution spectrogram may have difficulties dealing with these discontinuities.

We present two methods for energy normalization. The first one uses Parseval's identity as a theoretical energy reference. Parseval's identity can be written in the following way for the DFT [BB09]:

$$\|y\|^2 = \frac{\sum_i \sum_j a_{i,j}^2}{N}$$

where y is a signal in the time domain, $a_{i,j}$ are its DFT coefficients and N is y's length in samples. Given a subregion of signal y defined by a time and frequency range, we use Parseval's identity to calculate its energy reference in the following way:

1. Slice y in the specified time range;
2. Compute its DFT spectrum and select the positive coefficients of frequencies inside the specified frequency range;
3. Calculate the sum of the squared coefficients, multiply this by two (in order to consider the symmetric negative coefficients) and divide it by N.

When computing a localized STFT, we apply a normalization factor, matching its energy sum with the energy reference taken from Parseval's Identity. By doing this to all spectrograms that are part of the IRMS, we guarantee that their energies are normalized using the same theoretical reference value originated from the time-domain signal. Although this method is strongly supported in a mathematical basis and produces good results, it is also cost-intensive, since we have to perform a DFT for all refined subregions. As a cheaper alternative with similar results, we also present a peak-matching strategy.

In the peak-matching strategy, we use as reference the initial rough spectrogram of the IRMS. When inserting a localized STFT on top of it, we check the highest peak value of the corresponding subregion of the initial spectrogram and normalize all values of the spectrogram being inserted according to the original highest peak. This idea extends to the multilevel IRMS: when inserting a level 2 STFT into a level 1 STFT, we match the peak of the level 2 STFT to the peak of the corresponding subregion from level 1. Surprisingly, this approach seems to create a more uniform-looking (in terms of noise level across different subregions of the TFP) spectrogram than the Parseval approach, while adding no significant computational cost to the single-level IRMS. Fig. 4.9 shows a comparison between the IRMS with no normalization and both presented normalization strategies.

Although the peak-matching strategy creates a more uniform-looking spectrogram, it should not be considered the "best" normalization strategy, since this evaluation is highly dependent on the final use of the IRMS. Even for visualization purposes, one could argue that the un-normalized IRMS, by presenting a lower noise level and more concentrated peaks in the refined subregions, is actually focusing on the most relevant elements of the TFP, making notes, harmonics and onsets easier to inspect. Plus, many MIR algorithms using the IRMS as input representation could be computed locally within each subregion with fixed resolution, thus avoiding the discontinuity issue and taking advantage of the lower noise level in refined subregions. With this in mind, the implementation leaves the peak-matching normalizaton as a user-defined parameter of the IRMS, to be determined according to its final use.

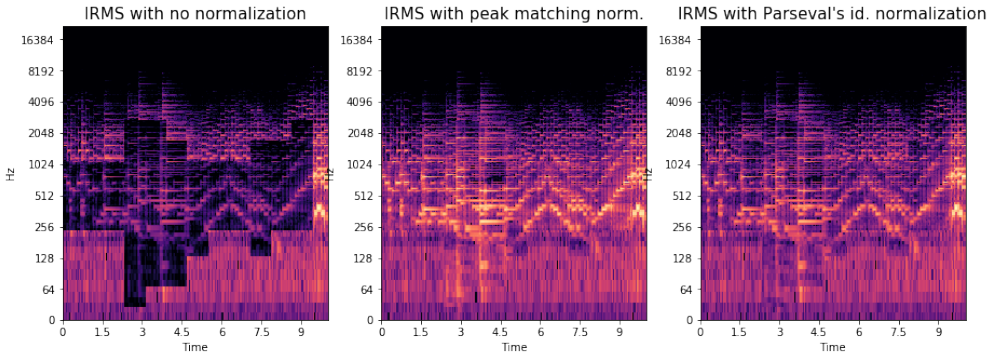


Figure 4.9: *A comparison between the IRMS with no normalization and the two tested approaches. The lack of normalization causes a severe discontinuity in the edges of refined subregions, while the peak matching strategy results in a smoother transition between all areas of the TFP.*

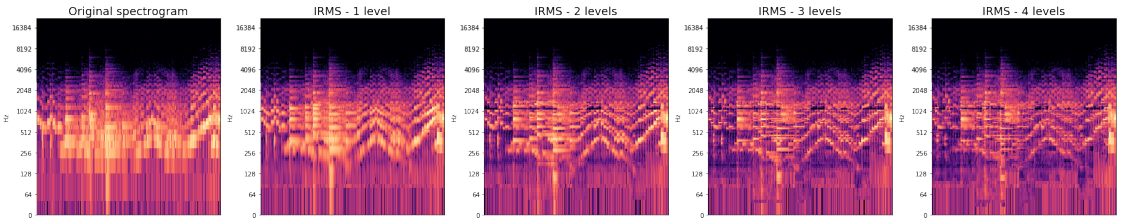


Figure 4.10: *The multilevel IRMS. Parameters used were, for refinement levels 1 through 4 respectively: k factors of 2, 4, 8 and 16; subregion sizes of 1600x1600, 800x800, 400x400 and 200x200 (cents × ms); refinement percentages of 35%, 75%, 75% and 75%.*

Multilevel considerations

We presented the algorithm focusing on the single-level IRMS. Minor changes occur in its multilevel configuration. Some of the user-defined parameters need to be supplied per level: subregion size, detection percentile/threshold and the k factor (it is necessary that the subregion size defined for level $l+1$ is smaller than that of level l). After the first level of refinement, a for-loop iterates through the remaining levels defined by the user. For level $l > 1$ of refinement, the algorithm takes the following steps:

1. We select from the IRMS data structure all subregions refined in level $l-1$;
2. For each of these subregions, we repeat the detection and refinement stages of the algorithm with the user-defined k factor, subregion size and detection percentile/threshold for the current level;
3. All newly-refined subregions are normalized and inserted into the IRMS data structure.

At the end of the last level defined by the user, we return the IRMS data structure to the user. The functions and algorithms for detection and refinement in levels $l > 1$ are the same as the ones used in level 1, except that they take as one of their arguments a STFT matrix that does not represent the entire TFP, but only part of it. Fig. 4.10 shows IRMS visualizations for an increasing number of refinement levels.

4.2.2 Data structure

Storing a multiresolution spectrogram is not a trivial task. The final product of such a representation is a matrix with different time and frequency lattices, a 2-D image with varying pixel density throughout the time-frequency plane. This variation in density means it cannot be easily stored in a single matrix, and how we choose to do so could in the end undermine our efforts in

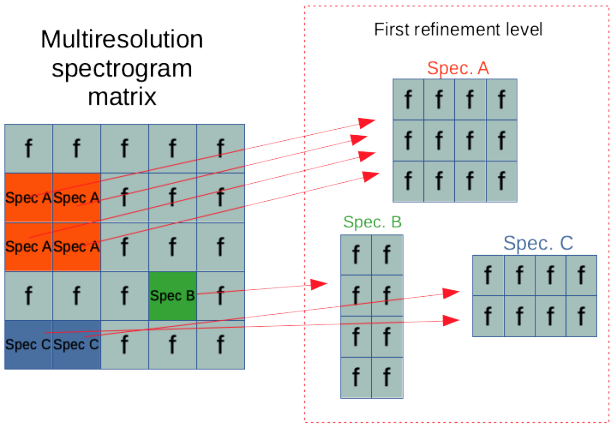


Figure 4.11: *A representation of the IRMS data structure for its single-level configuration. Each bin marked with an 'f' is part of a subregion that was not refined, and so it points to a float value representing its spectral amplitude. Bins that are part of refined subregions point to the corresponding new representation of its subregion.*

building an economical (memory-wise) representation. For instance, one simplistic solution would be to build a matrix that uniformly uses the maximum resolution present in the multiresolution spectrogram, interpolating or repeating values when necessary. While this would work in presenting an accurate visualization of the multiresolution spectrogram, it would not take advantage of the savings in memory space achieved by using lower resolutions in the coarser regions of the TFP. We first present our conceptual solution to this problem, then take a look at the actual classes and methods implemented in our algorithm.

Fig. 4.11 shows the concept behind the implemented data structure. It functions on the central idea of the IRMS: starting from a rough spectrogram and refining specific subregions of the TFP. Note that a refined subregion is thought of as a rectangular selection of bins from the initial spectrogram, that after being refined will be represented by a new collection of bins (a new STFT matrix). This means that we could substitute the original bins of a subregion for this newly calculated local STFT, but doing so inside the initial STFT matrix would cause problems because of the difference in density from the other subregions of the TFP. Our solution is to build a matrix of bins that fall under two cases: either they are part of a subregion of the TFP that was refined further or they are not. In the first case, this bin points to another matrix, one that represents the subregion from which it is part of in greater detail. In the second case, this bin is the most detailed representation of the subregion it represents, and so it will point to a float value representing its spectral amplitude.

Extending this idea to the multilevel case (Fig. 4.12), we end up with a tree-like structure with the following characteristics:

- Each node is a spectrogram. Each bin of a spectrogram can either point to a float or another node of the tree;
- Its root is the initial rough spectrogram;
- The nodes of level l of the tree are the spectrograms computed in refinement level $l+1$ of the IRMS.

In order to implement this data structure, two custom classes were created:

- `SingleResSpectrogram`: an object that stores a single resolution spectrogram, with the following attributes:
  - `spec`: a 2-D matrix that stores an STFT spectrogram;
  - `x_axis`: center times of the `spec` matrix time frames;

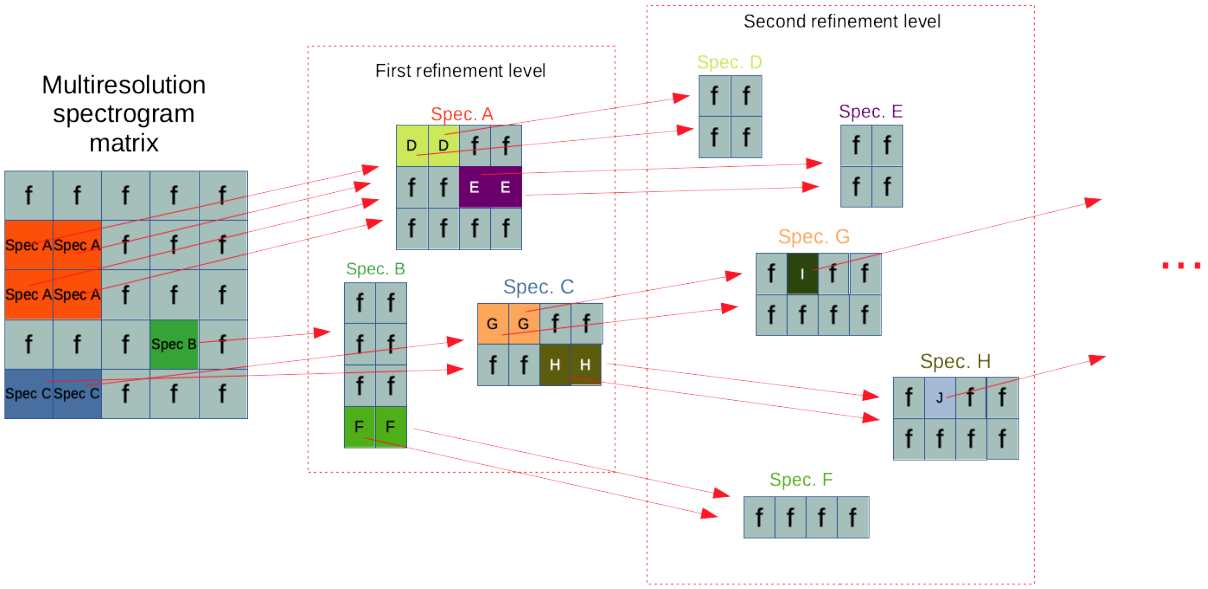


Figure 4.12: *A representation of the IRMS data structure for its multi-level configuration. The expansion of the original idea to multiple levels of refinement creates a tree-like structure, where each level of the tree corresponds to the spectrograms of each refinement level.*

  - `y_axis`: center frequencies of the `spec` matrix frequency bins;
  - `n_fft, hop_size`: the values used in the calculation of the spectrogram
  - `sr`: sampling rate of the signal from which the STFT was computed.

- `MultiResSpectrogram`: an object that stores a multiresolution spectrogram, with the following attributes:
  - `base_spec`: this attribute is initialized with a `SingleResSpectrogram` that stores the initial rough spectrogram that will be refined, and is the root of the tree data structure;
  - `zoom_specs`: this is an auxiliary list for the computation of a visualization of the IRMS. Its i-th element contains the list of spectrograms computed in level i of refinement, each represented by a `SingleResSpectrogram` object.

Insertion

When a subregion is refined, a `SingleResSpectrogram` object is produced with the localized STFT as its `spec` attribute. We then insert this object into the IRMS data structure. This is done by pointing some of the bins of its parent spectrogram - the spectrogram from which this subregion was detected as musically relevant - to this newly refined `SingleResSpectrogram`.

We need only to define which bins will be pointed towards this new STFT. Logically, these are the bins that form the subregion that was detected, and is now being represented with this localized spectrogram. However, the sub-band algorithm uses safe-guards to produce an STFT that represents a slightly wider frequency band than the one occupied by the original detected subregion. Because of the steps involved in the sub-band algorithm, the center frequencies of the localized STFT do not line up with the center frequencies of the original STFT. The insertion algorithm errs on the side of "over-representation": when a subregion's frequency frontier falls inside a bin of its refined STFT, we include this "part-in part-out" bin in the matrix instead of leaving it out. In short, we are actually pointing to an initial subregion into a representation of a slightly bigger subregion (Fig. 4.13 illustrates this characteristic). This is not necessarily a problem, given that all bins pointing towards this new STFT are represented in it, which is what we would expect from a look-up operation.

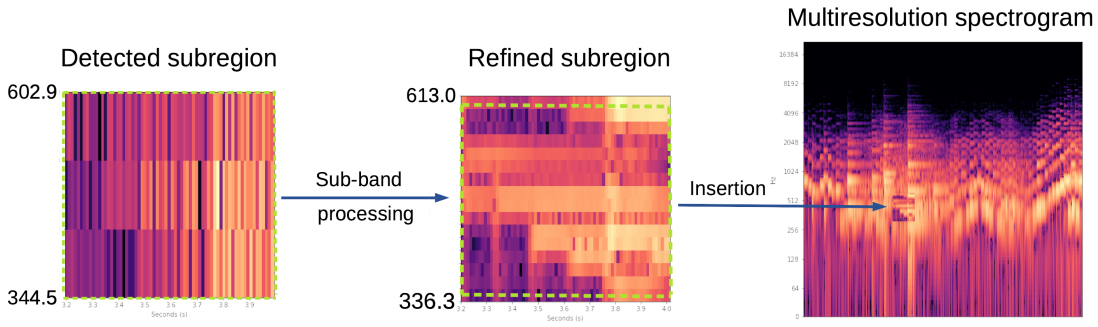


Figure 4.13: *A visualization of the insertion procedure in the IRMS data structure. When a detected subregion (here pictured as the subregion between 344.5–602.9 Hz and 3.2–4.0 seconds) is refined, we create a representation of a slightly larger frequency band (here pictured as 336.3–613.0 Hz). The dotted green lines in the second plot show the bounds of the original detected subregion. In the final step, all bins of the original spectrogram inside this dotted green area point towards the refined subregion. This means that an area of bins from 344.5-602.9 Hz will point to a representation of a 336.3–613.0 Hz frequency band.*

Given a new `SingleResSpectrogram` to be inserted and its parent `SingleResSpectrogram`, we execute Algorithm 2. It calculates the indices that define the appropriate bins of `parent_spec` to be pointed at `new_spec`, normalizes the spectrogram to be inserted, inserts it and also updates the auxiliary `zoom_specs` structure.

Algorithm 2 Function insert_spec(new_spec, parent_spec, normalize=True):

$x_{start} \leftarrow$ the element of parent_spec.x_axis that is nearest to (and greater than) the first element of new_spec.x_axis
$x_{stop} \leftarrow$ the element of parent_spec.x_axis that is nearest to (and less than) the last element of new_spec.x_axis
$y_{start} \leftarrow$ the element of parent_spec.y_axis that is nearest to (and greater than) the first element of new_spec.y_axis
$y_{stop} \leftarrow$ the element of parent_spec.y_axis that is nearest to (and less than) the last element of new_spec.y_axis
new_spec $\leftarrow$ normalize(new_spec, parent_spec)
parent_spec.spec$[y_{start} : y_{end}, x_{start} : x_{end}] \leftarrow$ new_spec
append new_spec to the appropriate element of IRMS zoom_specs

Visualization

So far, we have presented a data structure capable of economically storing a time-frequency multiresolution representation. We still have to deal with how to generate a visualization of this structure. We do this with an analogous algorithm to that of insertion, but instead of inserting `SingleResSpectrogram` objects, we paste 2-D images on top of each other. Algorithm 3 describes the pseudo-code for this operation. It uses the auxiliary list `zoom_specs` from the `MultiResSpectrogram` object as a quick shortcut to retrieving each localized STFT from each refinement level. The algorithm goes through this list, pasting each localized STFT into the appropriate subregion of the original STFT. We make sure to access the list in correct order, so that images from level l are always pasted on top of images from level $l-1$ and not the other way around.

This algorithm results in a 2-D image to be displayed to the user. Note that we start by expanding the initial spectrogram to the dimensions of the final image that will be presented to the user. This means that the actual resolution of the spectrogram visualization depends on these dimensions, and if we want to display the resolution found on the most refined subregions, we end up having the problem cited on the beginning of this section: thinking of a spectrogram bin as a pixel, in order to display the full information present in a refined subregion, we would have to

Algorithm 3 Function generate_visualization(irms):

```
spec_img ← irms.base_spec resized to the dimensions of the image displayed to the user
for each level in irms.zoom_specs do
  for each local_spec in level do
    box ← determine_bounds(local_spec, spec_img)
    spec_img ← paste_img(spec_img, local_spec, box)
  end for
end for
return spec_img
```

expand all other subregions to the same pixel density, i.e. the same frequency and time resolutions. This expansion is done with nearest-neighbour interpolation, a method in which the value of a pixel in the resized image is calculated as the value of the nearest pixel of the original image. That is, we are not creating values that do not exist in the original STFT matrix, but only repeating those that were already there. This ensures that every pixel in the visualization is interpretable as spectral information of the original signal, and not as a result of interpolation.

Although this expansion is an expensive and inefficient operation, this is done for visualization purposes only. The IRMS is stored in a compact form, and algorithms that use the IRMS as their input can work locally on the STFT matrices that form its data structure. Future work can include more sophisticated image processing tools for the display of an image with non-uniform pixel density, in order to speed up the generation of a visualization of the IRMS.

Future work on the data structure

Future work on this data structure includes the development of a look-up method that is capable of, given a frequency and time value, returning the most detailed bin that represents this region. We should also think of a way of indexing bins so that the IRMS tree structure can be accessed as a simple 2-D matrix, which would facilitate its use with ready-to-use MIR algorithms. Lastly, in order to generate its visualization without the auxiliary `zoom_specs` list, future work should include the development of a fast method for the retrieval of all matrices of a given refinement level from the data structure.

4.2.3 Complexity analysis

In this section, we offer a complexity analysis of the IRMS, focusing on the computation of STFT representations for each TFP subregion of the initial spectrogram. For a step-by-step execution time profiling of the IRMS, please refer to Appendix 6.2.

The IRMS's initial STFT has a complexity of

$$\mathcal{O}(m \cdot \log(n));$$

where m is the total number of samples of the analyzed signal and n is the number of rows of the STFT matrix. Assuming a worst case in which all subregions of this initial spectrogram are detected as musically relevant, we need to perform the defined sub-band processing algorithm for the refinement of each one.

All steps of the sub-band algorithm up to the final localized STFT (filtering, subsampling and ring modulation) are linear in cost, and thus will be absorbed by the cost of the final STFT. We then only need to analyze the cost of performing an STFT for each TFP subregion of the initial spectrogram. We define a subregion size by its width m' in samples and its height n' in number of rows of the initial spectrogram, with one caveat: since in our implementation the subregion size is defined in ms per cents, the number n' of rows is not fixed, but changes along the frequency axis. For simplicity sake, we perform our analysis using octave frequency bands, i.e. a subregion height of

1200 cents. For different subregion heights, we would have an analogous analysis with proportional cost, which would result in the same computational complexity.

For the analysis of a single slice of m' samples, we have a subregion of height equal to $n/2$ bins in the highest octave, followed by a subregion of height equal to $n/4$, $n/8$ and so on. Considering the refinement parameter K that defines the number of rows of the localized STFT as a function of the subregion height in bins, the STFT cost of all of these stacked subregions summed is given by:

$$\mathcal{O}\left(m' \log \frac{Kn}{2} + m' \log \frac{Kn}{4} + \cdots + m' \log \frac{Kn}{2^L}\right)$$

where $L = \log Kn$. Additionally,

$$\begin{aligned} \log \tfrac{Kn}{2} + \log \tfrac{Kn}{4} + \cdots &= (\log Kn - \log 2) + (\log Kn - \log 4) + \cdots + (\log Kn - \log 2^L) \\ &= L * \log Kn - \log 2 * (1 + 2 + 3 + \cdots + L) \\ &= L^2 - \log 2 * \tfrac{L(L-1)}{2} \end{aligned}$$

Both terms above are proportional to $L^2 = (\log Kn)^2$, and so the total cost of all subregion STFTs in a vertical slice of the initial STFT is given by:

$$O(m'(\log Kn)^2)$$

Now, considering we have a total of $\frac{m}{m'}$ vertical bands in the initial spectrogram, the IRMS complexity can be expressed by:

$$O(m(\log Kn)^2)$$

4.2.4 Code organization

Code is available on GitHub, and is divided into the following files:

- `irms.py`: contains the entry point function `irms`, that should be called by the user for the computation of a IRMS representation. Its parameters are:
  - **y**: the array containing the audio signal to be analyzed;
  - **k**: the k factor (either a single value or a list, one for each level of refinement);
  - **subregion_size**: subregion size for detection and refinement (either a single value or a list, one for each level of refinement);
  - **model**: a scikit-learn predictive model to be used to calculate probabilites for the refinement of a subregion;
  - **pct**: detection percentile (either a single value or a list, one for each level of refinement). Alternatively, the function can receive a probability threshold instead of a percentile;
  - Optional parameters (default values are given): `n_fft=512` and `hop_size=512` for the initial spectrogram, and `sr=44100` (sampling rate of the y signal).
- `mappings.py`: contains all functions pertaining to the extraction of features from a spectrogram. Its main function is `extract_features`, that partitions a spectrogram into subregions and extracts the Rényi and Shannon entropies from each one. It accepts the following parameters:
  - **spec**: the amplitude spectrogram matrix from which the features will be extracted. In practice, a dB energy spectrogram will be derived from this one, from which the Shannon entropy will be extracted, while the Rényi entropy will be extracted from the original amplitude spectrogram;

  - **subregion_size**: subregion dimensions given in cents per ms;
  - **fft_freqs**: optional, used when features are being extracted from a subregion spectrogram. Contains the y-axis center frequencies of the spectrogram;
  - Optional parameters (default values are given): `n_fft=512, hop_size=512` (window and hop size of the analyzed spectrogram), `sr=44100` (sampling rate of the original signal), `alpha=3` (the α factor used in the calculation of the Rényi entropy).

- `detect_musical_regions.py`: contains all functions that relate to the detection of musical subregions of a spectrogram. Functions from `mappings.py` are called here for feature computation. Its main function `detect_musical_regions` takes a spectrogram, a predictive model and a subregion size as its parameters and returns a list of indices representing the detected subregions. These are transformed by `musical_regions_to_ranges` into a list of `time_range, freq_range` pairs that represent the detected subregions.

- `stft_zoom.py`: contains all functions related to the sub-band processing algorithm. Its main function, `stft_zoom`, takes these inputs:

  - **y**: an array containing an audio signal;
  - **freq_range**: a list of two values that define a frequency band;
  - **time_range**: a list of two values that define a time slice;
  - **k** and **original_window_size**: these two parameters define the frequency resolution of the final STFT;

  and returns an STFT computation of y in the TFP subregion defined by freq_range and time_range.

- `classes.py`: contains the definitions for the data structures `SingleResSpectrogram` and `MultiResSpectrogram`;

- `util.py`: other utility functions.

The following Python packages were used (all of which are available with `pip install`): LibROSA for the STFT computations, SciPy for the filtering stages of the sub-band algorithm and Shannon entropy calculation, fast_histogram for the histogram part of the entropy calculation, Pillow for the generation of the `MultiResSpectrogram` visualization and NumPy for general array computations.

Chapter 5

Experiments

This chapter presents three experiments that form the proposed methodology for the development of an efficient multiresolution representation. Sections 5.1 and 5.2 analyze two experiments related to the evaluation of musical information estimators of the TFP, while Sec. 5.4 evaluates the final IRMS algorithm with a comparative experiment based on execution time.

Sections 5.1 and 5.2 are heavily based on [FQ20], where these two experiments were originally presented.

5.1 Estimator evaluation

This experiment aims to evaluate different features as musical information estimators in the TFP. When extracted from a subregion of a spectrogram, a good estimator should give us an indication of whether the this subregion contains musical events (notes, harmonics, expressive elements) or not. The evaluated estimators and reference values were detailed in Secs. 3.3.1, and now we focus on the technical aspects of the experiment as well as the results and their discussion.

5.1.1 Experiment

The MAESTRO dataset [HSR+19] was used in this experiment. It contains over 200 hours of piano performances recorded in Yamaha Disklaviers as part of the International Piano-e-Competition, spanning ten years of competition. It should be noted that this dataset is mostly comprised of pre-20th century western-european classical music, which limits our study towards this style of music and the musical characteristics and expressiveness of the piano. For each performance, a pair of a MIDI file and sound recording is captured and aligned with 3ms accuracy. From this dataset, we randomly selected 16 minutes of performances from 2004 to 2014. In order to represent different recording conditions in the 16 minutes of music selected for the experiment, each year in the dataset was sampled equally. For every recording/MIDI pair selected, 30 seconds were extracted from the halfway point of the performance and used in the experiment.

Each selected performance was used in the following analysis: from the corresponding MIDI file, the three reference maps described in 3.3.1 were extracted, according to a rectangular subregion size: piano roll, piano roll with harmonics, and piano roll with harmonics and decay model. For modeling the energy decay over time and frequency of the third reference value, we used models based on measured data from acoustic pianos [CDM15, Bla65]. For each note, a decay of 8 dB/s is considered, and seven harmonics are included with a decay of 4.3 dB per partial. MIDI velocity values in [CDM15] were translated using estimates from [BFS+02], as decays of 28 velocity points per second and 7.86 velocity points per partial. From the corresponding audio recordings, an STFT analysis was performed, from which the estimators described in 3.3.1 were extracted (Rényi entropy, Shannon entropy, standard deviation and mean value of amplitude, energy and dB energy spectrogram - 12 estimators in total), using the same subregion size used for the extraction of the ground truth values. The Rényi entropy was computed with $\alpha = 3$, a value justified by the discus-

Estimators	**Piano roll**	**Piano roll + harmonics**	**P. roll + harm. + decay**
Rényi (amp. spec)	0.43 +- 0.08	0.51 +- 0.09	0.49 +- 0.09
Shannon (dB spec)	0.26 +- 0.05	0.34 +- 0.06	0.29 +- 0.05
Std. dev. (amp. spec)	0.51 +- 0.10	0.55 +- 0.10	**0.64 +- 0.08**
Std. dev. (en. spec)	0.36 +- 0.10	0.39 +- 0.10	0.50 +- 0.09
Density (amp. Spec)	**0.55 +- 0.12**	**0.57 +- 0.14**	**0.66 +- 0.11**
Density (dB spec)	0.48 +- 0.10	0.52 +- 0.12	0.51 +- 0.11

Table 5.1: *Correlation results of the best performing estimator/reference value pairings, using a subregion size of 800 ms per 800 cents and an STFT with an analysis window of 2048 samples. All correlations achieved significance values $p < 0.05$. Reproduced from [FQ20].*

sion in [BFJM01]. Finally, the Pearson correlation between each possible estimator/reference value pairing (36 pairs in total) was computed.

In order to test the influence of the resolution of the STFT from which the estimators are extracted, the experiment was repeated for spectrograms computed with windows of 512, 1024, 2048 and 4096 samples. All recordings are sampled at 44100 kHz, and hop sizes were chosen as one-fourth of the size of the analysis window.

As discussed in Sec. 3.3.1, the dimensions of the subregions determine our reference value, and should be considered part of our definition of musical relevance. Since our work is strongly motivated by AMT, the best subregion size would be the one that produces the representation best suited for this task. Admittedly, even this motivation does not entail a single optimal subregion size: this would vary according to spectral characteristics of the audio being transcribed, such as the expected number of notes per second or the proximity in the frequency axis of simultaneous notes and harmonics per chord, so it is not appropriate nor possible to treat the subregion size as a variable to be optimized. With this discussion in mind, the experiment was repeated for subregion sizes of 100 ms by 100 cents, 200 ms by 200 cents, 400 ms by 400, 600 ms by 600 cents, 800 ms by 800 cents and 1000 ms by 1000 cents, in order to observe, compare and discuss the results of the obtained representations under different conditions.

The experiment was written in Python, using LibROSA [MMB+19] for the STFT calculations, NumPy for general array computations, fast-histogram[1] and SciPy for entropy calculations and Mido[2] for MIDI manipulation.

5.1.2 Results and discussion

Overall, the estimators that achieved the highest correlations were the density and standard deviation of the amplitude spectrogram (see table 5.1). Density of the dB spectrogram and Rényi entropy of the amplitude spectrogram also achieved fair results, although no estimator achieved correlation coefficients significantly above 0.5. Estimators extracted from the energy spectrogram (not in dB) did not achieve notable results, as well as Shannon entropies, that achieved the lowest correlation with each of the three reference values.

Fig. 5.1 shows that increasing subregion size tends to increase the correlation between reference value and estimator for all pairings. This is somewhat expected, given that using a bigger subregion size has the effect of making both matrices lose detail, favoring general trends in the data which are easier to estimate than minor local changes. No outliers were observed in this trend, which agrees with our discussion in 3.3.1 about the impossibility of finding an optimal subregion size. Further studies analyzing the impact of subregion size in a subsequent AMT task of selected pieces with similar spectral characteristics could provide more information about this behaviour.

The variation of correlation caused by STFT window size (see Fig. 5.2) seems to be fairly minimal for most estimators, although with some interesting exceptions and characteristics. The

[1] https://github.com/astrofrog/fast-histogram
[2] https://mido.readthedocs.io/en/latest/

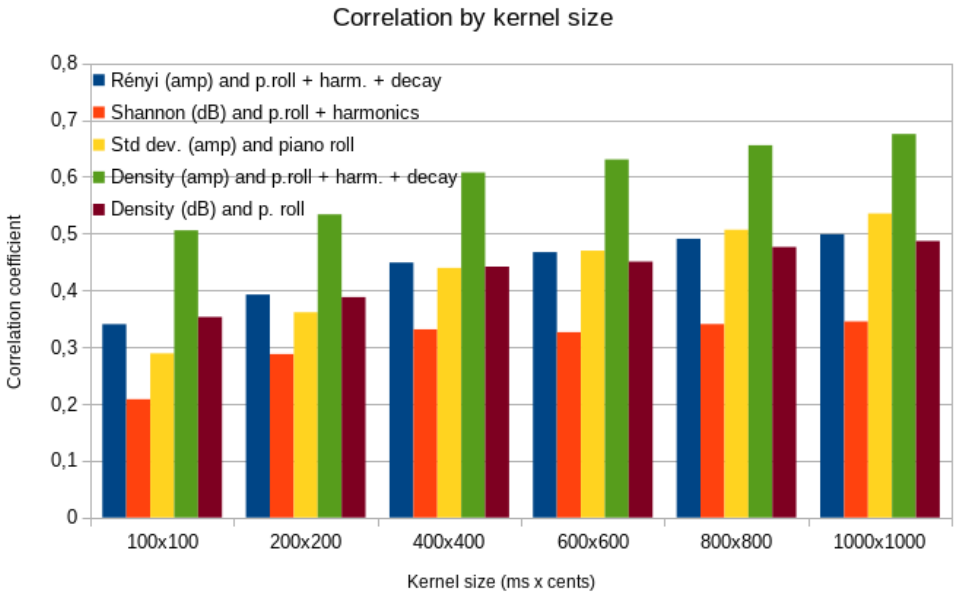


Figure 5.1: *The effect of subregion size in the correlation coefficients for 5 estimator/reference value pairings. All estimator/reference value pairings presented (approximately) monotonically increasing behaviour. Reproduced from [FQ20]*

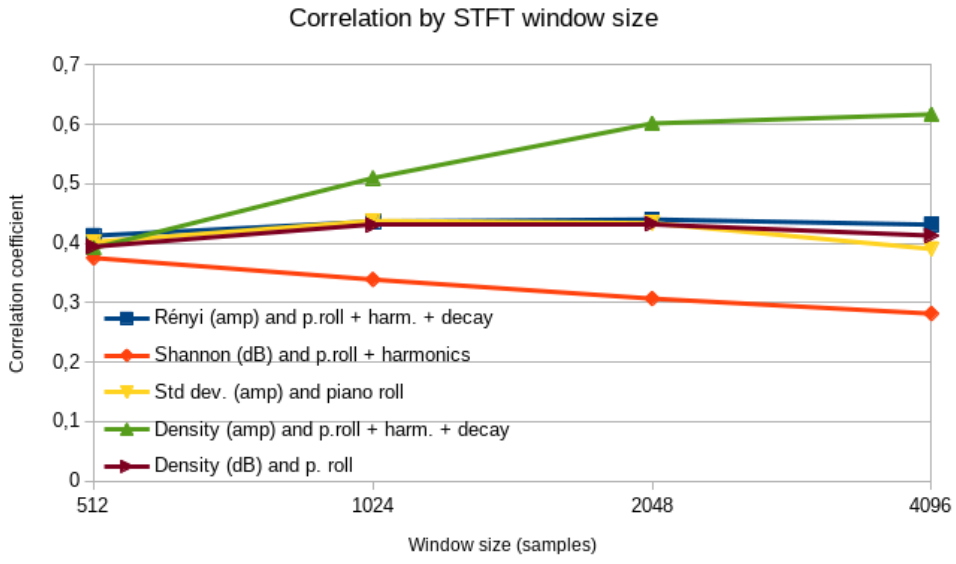


Figure 5.2: *The effect of window size in the correlation coefficients for 5 estimator/reference value pairings. Reproduced from [FQ20]*

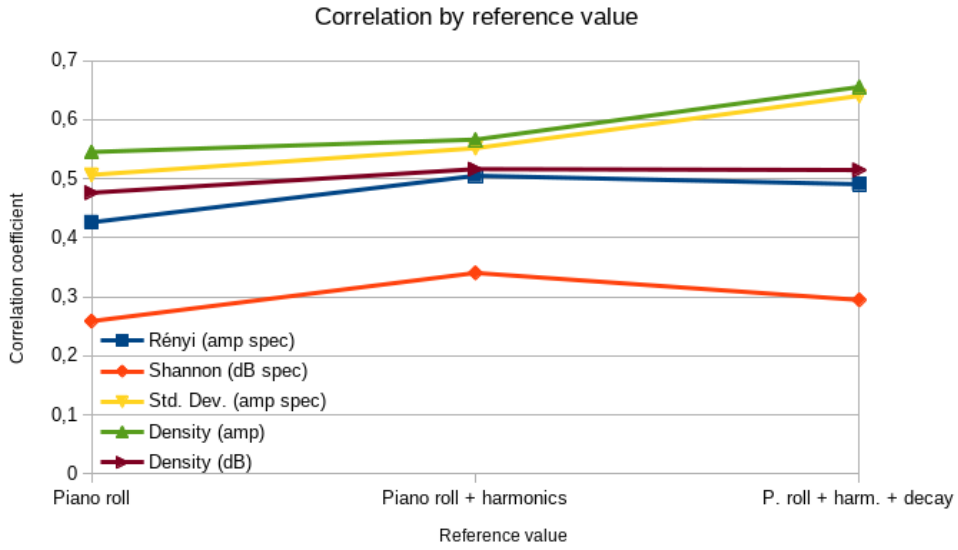


Figure 5.3: *The effect of introducing harmonics and decay models in the reference values for 5 estimator/reference value pairings. Reproduced from [FQ20]*

Shannon entropy is the only feature that presents monotonically decreasing performance with the increase of window size, while amplitude density seems to be the estimator mostly favored by an increase in window size. All other estimators are not as sensible to this variation.

Entropies in general performed better with the reference value of the piano roll with harmonics (see Fig. 5.3). This means that the introduction of decay models in the reference value actually served to (slightly) decorrelate these estimators with the reference. This could be related to the invariance of entropy with respect to data scaling, which pushes the reference and estimator maps in this case apart from each other as the reference decays. Standard deviation and amplitude density presented higher correlation with the reference value containing decay models, which agrees with its sensitivity to scaling and the adherence of the simple decay models to the amplitude behavior observed in the spectrograms. Surprisingly, the introduction of decay models did not improve the correlation of the density of the dB spectrogram and the reference value. This probably means that the linear decay models represent poorly the energy decay profiles exhibited in the dB spectrogram of the Disklavier recordings.

It is important to note the high variance of the obtained correlation values. Although there are noticeable and useful trends in the data, this fluctuation means that the estimator values should be further analyzed with caution in the context of relevant subregion classification, the performance of which is still unclear in the presence of these fluctuations. The binary classification experiment aims to assess this classification performance in the context of the iterative refinement of a multiresolution time-frequency representation.

5.2 Binary classification of TFP subregions

This experiment is a compliment to the estimator evaluation, where we take its best performing estimators and further evaluate them in the binary classification task of musically relevant subregions of the TFP.

5.2.1 Experiment

As in the previous experiment, the MAESTRO dataset was utilized. All recordings from 2004 to 2015 present in the dataset were utilized, totalling nearly 9 hours analyzed from 1043 recordings. Once again, for every selected recording/MIDI pair, 30 seconds were extracted from the halfway

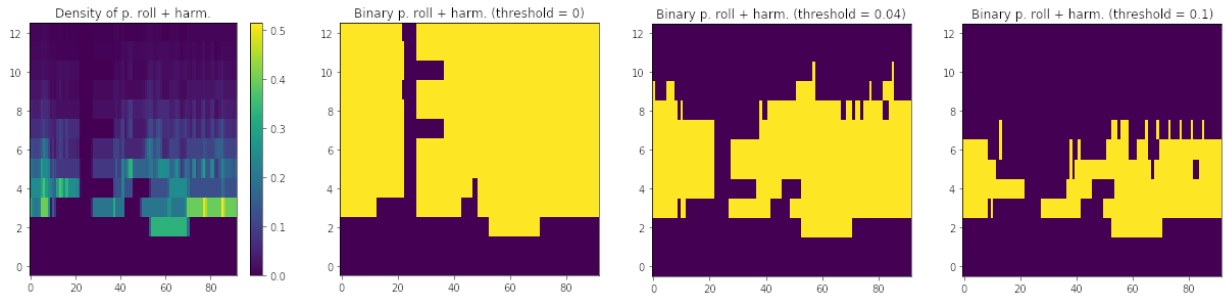


Figure 5.4: *The effect of the threshold in transforming the piano roll with harmonics reference value into a binary one. The piano roll with harmonics density map is shown, along with 3 binary maps obtained using thresholds of 0, 0.04 and 0.1 respectively. Reproduced from [FQ20]*

point of the performance and used in the experiment. 80% of the dataset was used for training and the remaining 20% for evaluation.

From the estimator evaluation experiment, the best performing estimators were selected, namely Rényi entropy, standard deviation and amplitude density of the amplitude spectrogram, and Shannon entropy and energy density of the dB spectrogram. An STFT window of 2048 samples was used, chosen as a middle ground motivated by the results shown in Fig. 5.2. A subregion size of 800 ms per 800 cents was used while keeping in mind that the utilized dataset contains piano performances with heterogeneous spectral characteristics, and it would not be possible to choose an ideal subregion size for this experiment.

The first step of this experiment consisted of the training of Naive-Bayes models for the prediction of the binary piano roll reference, using the mentioned estimators as single-feature models and every possible feature pairing as two-feature models. In a second step, designed to test the sensibility of each feature to the presence of harmonics, the models were evaluated using a binary representation of the piano roll with harmonics and decay models as ground truth. The rationale for also including the piano roll with harmonics and decay model in this experiment is not to evaluate the model for itself, but to gain insight on whether false positives of the binary piano roll model in step 1 could be related to the presence of harmonics, and also to enquire whether there are estimators that behave differently in the identification of subregions containing harmonics.

In order to transform the reference value explained in 3.3.1 into a binary map, a "musical density" threshold had to be chosen above which a bin was considered as a positive example of "musical activity". If this threshold is set to 0, the presence of even the weakest harmonic would signal a positive sample, and if it is set close to 1, only the "musically densest" regions would be counted as a positive sample (see Fig. 5.4). Several different thresholds were tested leading to the results presented in the following section.

In addition to the libraries cited in the previous experiment, scikit-learn [PVG+11] was used for training and evaluation of the Naive-Bayes models.

5.2.2 Results and discussion

Among the single-feature models for the binary piano roll prediction, the Rényi entropy performed best according to F-Score, achieving a recall of 0.78 and precision of 0.62. The Shannon entropy achieved a notable recall of 0.92 but a precision of 0.42. Among the feature pairings, Shannon entropy and standard deviation achieved the highest F-Score of 0.66, tied with the Rényi and energy density pairing. The highest precision of 0.76 was achieved by the Rényi entropy and standard deviation pairing, while the highest recall of 0.93 was reached by the pairing of Shannon and Rényi entropies. All notable results are shown in Table 5.2.

In order to interpret these results and what they mean for a possible multiresolution representation, we must first discuss the different implications of high precision and high recall in this setting. If a positive subregion of the TFP is expected to be represented in higher resolution by our

Estimator	F-Score	Precision	Recall	Avg. precision
Rényi (amp.)	**0.69**	0.62	0.78	0.69
Shannon (dB)	0.58	0.42	**0.92**	0.45
Std. dev. (amp)	0.52	**0.82**	0.38	0.72
En. density (dB)	0.65	0.61	0.69	0.67
Rényi + Shannon	0.62	0.47	**0.93**	0.67
Rényi + Std. dev.	0.63	0.76	0.53	**0.74**
Rényi + en. density	**0.66**	0.55	0.83	0.67
Shannon + en. density	0.63	0.49	0.91	0.66
Shannon + Std. dev.	**0.66**	0.73	0.59	0.72

Table 5.2: *Selected classification results for the trained Naive-Bayes models using the binary piano roll as ground truth. Reproduced from [FQ20].*

Estimator	Recall (w. harmonics)	Precision (w. harmonics)
Rényi (amp.)	0.82 (+0.04)	0.61 (-0.01)
Shannon (dB)	0.92	**0.60 (+0.18)**
Std. dev. (amp)	0.31 (-0.07)	0.84 (+0.02)
Amp. density (amp)	0.26 (-0.06)	0.81 (+0.03)
En. density (dB)	0.69	0.61

Table 5.3: *Classification results using the binary piano roll with harmonics (threshold set to 0.04) as ground truth. In parentheses the score change in relation to the piano roll model (without harmonics) using the same feature is shown. Reproduced from [FQ20]*

algorithm, false positives can be interpreted as spending computing power in unimportant regions, while false negatives are interpreted as withholding computing power in musically relevant regions that should be refined. Ideally, in a TFP representation aimed at AMT, we would like to represent in detail all regions containing musical information, even if this means spending a bit of computing power where this is not needed. Thus, when choosing between features with similar F-Scores, we favor the ones with higher recall over the ones with higher precision.

Several thresholds were tested for the introduction of harmonics in the ground truth label. A threshold of 0 leads to a percentage of 78% positive training samples in relation to all samples - in practice, nearly every region of the TFP above 200 Hz is labeled as positive for all training samples. When using the simple piano roll with no harmonics as ground truth, 30% of the training samples are labeled as positive. As a middle ground, a threshold of 0.04 was chosen (49% of positive samples). For this threshold, the performance of the Rényi and Shannon entropies improve (see Table 5.3), while standard deviation and amplitude density suffer a small drop in F-Score. The improvement in precision of the Shannon entropy model is of special importance: it means that some of the false positives detected by the model trained with the simple piano roll were actually regions occupied by harmonics, and thus regions that contained some musical information (which would be relevant e.g. to timbre analysis and melody extraction algorithms).

Taking all classification results and our discussion of precision and recall into consideration, both Rényi and Shannon entropies present encouraging results for their usage in an algorithm for producing a mutiresolution time-frequency representation for tasks such as AMT and timbre analysis, at least for musical pieces that follow the characteristics present in the utilized dataset. The pairing of both entropies, as well as the pairing of Shannon entropy and energy density could also be useful for this detection. These results also further validate the usage of Rényi entropy as a time-frequency information content estimator as seen in [BFJM01].

5.3 Further discussion about musical information estimators

The classification experiment concludes our investigation of possible estimators of musical information in the TFP, with promising results for the use of Rényi and Shannon entropies in the detection of musically relevant subregions of a spectrogram. In the end, the feature pairing of both entropies was chosen as the default for the detection phase of the IRMS, given its high recall and comparable F-Score with other tested models. It still remains unclear how the spectrogram's resolution from which these estimators are extracted influences the performance of detection. Since both entropies seem to perform best with a shorter window, a 512-sample window was chosen as the default for the IRMS, lewaving this decision open for the user.

Since the dataset used is comprised solely of piano performances, it would be important to evaluate how well these detection methods perform in other conditions and musical styles, such as performances of instruments with very different spectral characteristics from the piano and more complex multi-instrument performances. There is also interest in considering separately the frequency and time axes in our detection of relevant subregions in future work. Since the inherent tradeoff between time and frequency resolution in the STFT motivates our development of a multiresolution transform, a detection algorithm that distinguishes between regions that contain relevant time or frequency information could better guide the refinement step towards a more suitable representation for AMT, representing, for instance, onset regions with higher temporal resolution and melodic lines with higher frequency resolution.

5.4 Computational cost experiment

From the investigations on sub-band processing (Sec. 4.1), musical information estimators (Secs. 5.1 and 5.2) and the integration between the two (Sec. 4.2), a final algorithm for the IRMS was conceived. This multiresolution spectrogram has as one of its main objectives to be a more efficient adaptive representation than those following the traditional framework found in the literature (see 1.2). As a final experiment for the evaluation of the IRMS, we propose its comparison against several different fixed resolution, multiresolution and adaptive representations according to execution time, in order to assess if the implemented representation is indeed as efficient as intended.

5.4.1 Experiment

The MAESTRO dataset was once again used for the computational cost experiment. As in the classification experiment, all sound recordings from 2004 to 2015 present in the dataset were utilized. For each file, 30 seconds were extracted from the halfway point of the performance and used in the experiment, totalling 8.5 hours analyzed from 1023 different performances.

For each audio recording selected, the following representations were computed: STFT, CQT [SK10], SWGM [MdVB17], Lukin & Todd's maximal energy compaction principle (noted throughout as simply Lukin-Todd) [LT06], SLS [dC20] and several configurations of the IRMS (see 3.4.1 for more details on the chosen representations). Each representation was computed for target frequency resolutions from 86.13 Hz/bin to 1.35 Hz/bin, equivalent to analysis windows of $2^9 \dots 2^{15}$ samples on a 44100-Hz sample rate signal. Time-wise, all representations are computed with a fixed hop size of 512 samples, producing a homogeneous time resolution of 12 ms. As noted in 3.4.1, the representations considered here deal with frequency resolutions in different ways, so these target frequency resolutions have to be translated to specific representation parameters. The evaluated representations and their parameters are:

- STFT: the analysis window size is chosen according to the desired resolution;
- CQT: the resolution in bins per octave of the CQT is set as the number of bins that fit into the octave from 100 to 200 Hz using the target frequency resolution in Hz per bin. Other parameters used for the CQT are a minimum frequency of 20 Hz and a range of 10 octaves;

- Lukin & Todd, SWGM and SLS: these three methods combine previously computed fixed-resolution representations, so their maximum frequency resolution is determined by the finest (frequency-wise) representation in the provided dictionary. For this group of representations, given a frequency resolution point defined by window size w_{max}, three STFT spectrograms are computed, with windows $w_{max}/8$, $w_{max}/2$ and w_{max}. Other parameters include a sparsity analysis window of 5 time frames per 48 frequency bins and energy attenuation window of 5 time frames per 16 frequency bins for the SLS (as per discussion in [dC20]), and a sparsity analysis window of the same dimensions for the Lukin-Todd representation.

In order to observe the influence of subregion size and number of refinement levels in the execution time of the IRMS, several configurations were evaluated. They all start with an initial spectrogram computed with an analysis window of 512 samples and use subregions of N cents per N ms, being divided into categories according to the number of refinement levels:

- Single level of refinement: subregion sizes with N = 1600, 800, 500 and 200 were tested. The target frequency resolution is attained at the refined subregions, by setting the k parameter accordingly. This parameter determines a multiplying factor in relation to the initial spectrogram: for example, for an initial spectrogram computed with a 512-point window and a target resolution equivalent to a 4096-point window, k is set to 8.

- Multilevel: 2 to 4 levels of refinement were evaluated, using subregion sizes with N = 1600, 800, 400 and 200.

In the multilevel configurations, the target frequency resolution is reached only by the refined subregions of the last level. The k factors for these configurations are set like so: defining $k^i_{l,res}$ as the k factor of the i-th level of the configuration with a total of l levels and a desired resolution res, it can be calculated as a function of $k^1_{1,res}$, by the formula $k^i_{l,res} = \frac{i}{l} \cdot k^1_{1,res}$. In concrete terms, this means that in multilevel configurations refinement occurs gradually, starting at 86.13 Hz/bin (initial spectrogram) and working its way up to the target resolution with uniform steps at each level.

Time resolution is kept constant (512 samples) across all subregions of the IRMS, as required by the experiment definition. Since subband processing includes a subsampling or undersampling step, hop sizes of the STFTs of refined subregions are redefined to preserve time resolution. Although this time resolution could be excessive for note onset tasks (note onsets are tipically considered correctly labeled if within 50ms of the ground truth [Dix06]), it was set as a common point between all evaluated representations in order to create a fair comparison.

One last parameter of the IRMS remains: the percentage of refined subregions on each level of refinement. This parameter was estimated for each audio file based on piano roll representations obtained from the paired MIDI files in the MAESTRO dataset. This is done by examining the corresponding MIDI file and extracting the percentage of subregions that contain at least one note (Figs 5.5 and 5.6 illustrate this process). For the multilevel configurations, we compute this reference for each level, by calculating the TFP area occupied by relevant regions on the current level and dividing it by this same measurement of the preceding level (that is, how much of the previously defined relevant subregions remain relevant in this deeper level). This strategy was adopted to ensure that the target frequency resolution was attained not only within the regions of the TFP that are musically most relevant, but within *all* regions that include music events in the piano roll. Since these reference values tell us how much of the TFP is usually occupied with notes, they can be useful later on for setting a default parameter for the IRMS for trascription related tasks (see 5.4.3 for further discussion on this). In order to assess a worst-case scenario related to refinement percentage, we also tested a "stressed" version of the IRMS, using a subregion size of 800 ms per 800 cents and the maximum refinement percentage found from the MIDI files in the MAESTRO dataset for this subregion size.

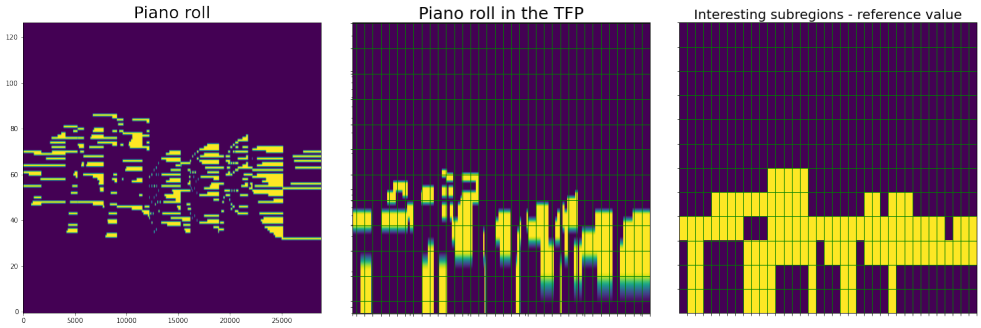


Figure 5.5: *Extraction of a reference for the refinement percentage of the single-level IRMS. Given an audio file, we take the corresponding MIDI file in the MAESTRO dataset, and build its piano roll. Then, we transform this piano roll into TFP dimensions, and divide the plane into subregions defined by a certain size in cents per ms (this same size will be used in the IRMS). Finally, we compute the ratio between subregions that contain at least one note and the total number of subregions, and use this as the refinement percentage of the IRMS. The green lines are the borders of each subregion.*

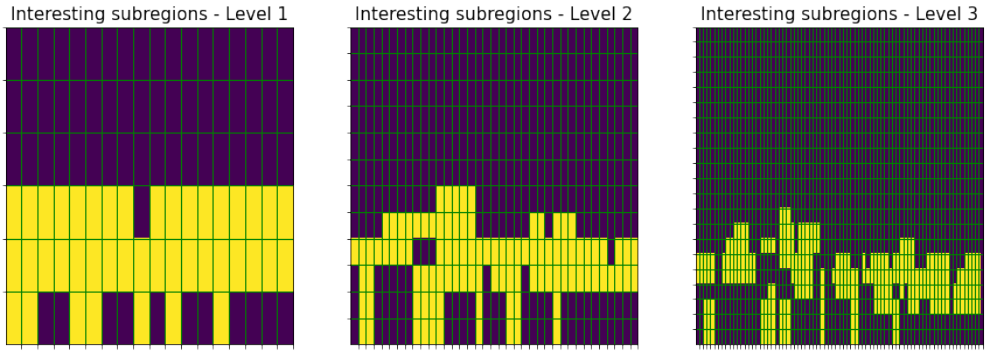


Figure 5.6: *Extraction of refinement percentages for the multi-level IRMS. The three figures present the final product of the process shown in Fig. 5.5 for three different increasingly smaller subregion sizes related to each refinement level. Refinement percentages for levels $l > 1$ are calculated as the ratio between the percentage of area occupied by relevant subregions (yellow area) for level l and this same measurement for level $l - 1$.*

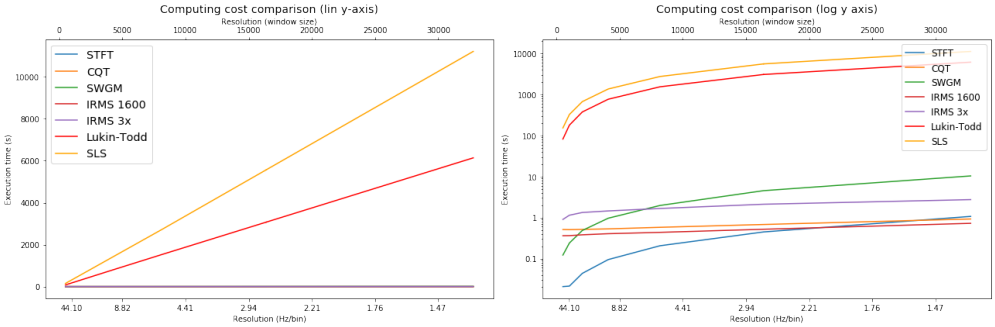


Figure 5.7: *Computational cost curves for all evaluated representations and selected configurations of the IRMS, with a linear y-axis (left) and logarithmic y-axis (right).*

In order to minimize the influence of other processes running on the same machine simultaneously with the experiment, each file was processed 5 times. We then take the mean of all computations of each representation and resolution pairing in order to plot the execution time curves.

5.4.2 Implementation details

This experiment was written in Python. Librosa [MMB+19] was used for the computation of STFT and CQT, and custom implementations of the Lukin & Todd, SWGM and SLS representations were developed. In the case of Lukin & Todd, no implementations from the authors were available; for the SWGM and SLS, MATLAB implementations were supplied by the original author. In the end, in order to maintain a common ground of coding language and runtime environment, the MATLAB codes were translated to Python with the supervision of the original author. This translation is also part of an ongoing collaboration with the original author on a Python package containing the proposed representations in [dC20] and other adaptive representations.

General experiment code and representation implementations rely on NumPy for array computations and SciPy for the computation of Hamming windows for the SLS. Execution times are captured using the `default_timer` function of the `timeit` Python module. This experiment was conducted on a 2017 MacBook Pro with 8 GB of RAM and a 3.1 GHz Intel Core i5 Dual-Core processor.

5.4.3 Results and discussion

Fig. 5.7 shows computational cost curves for all representations and two configurations of the IRMS (in the presented figures, single-level IRMS configuration with a subregion size of N cents per N ms are labeled as IRMS N, while configurations with L levels of refinement are labeled as IRMS Lx). The Lukin-Todd and SLS computational costs sit several orders of magnitude above the rest. These representations start with the computation of three different spectrograms, then interpolation of the smaller spectrograms to the dimensions of the largest matrix in the dictionary, and then looping through each bin of each spectrogram while calculating a measure of sparsity. Both measures of sparsity depend on selecting a subregion around the analyzed bin and then sorting all magnitude values therein, which amounts to a comparatively very high cost. Because of this order-of-magnitude difference, all subsequent figures only display the computing times for the 4 remaining representations, which are computationally more affordable.

Fig. 5.8 shows the computational cost curves of the SWGM, STFT, CQT and two configurations of the IRMS. Apart from the multilevel IRMS, all other representations seem to behave almost linearly in the analyzed interval. The graph shows that for resolutions finer than 4.41-Hz per bin the SWGM representation is no longer competitive with the others. These results, in conjunction with Fig. 5.7, show that the use of adaptive representations that rely on the traditional framework presented in [LT06] is impractical when dealing with large datasets and resolutions finer than 4

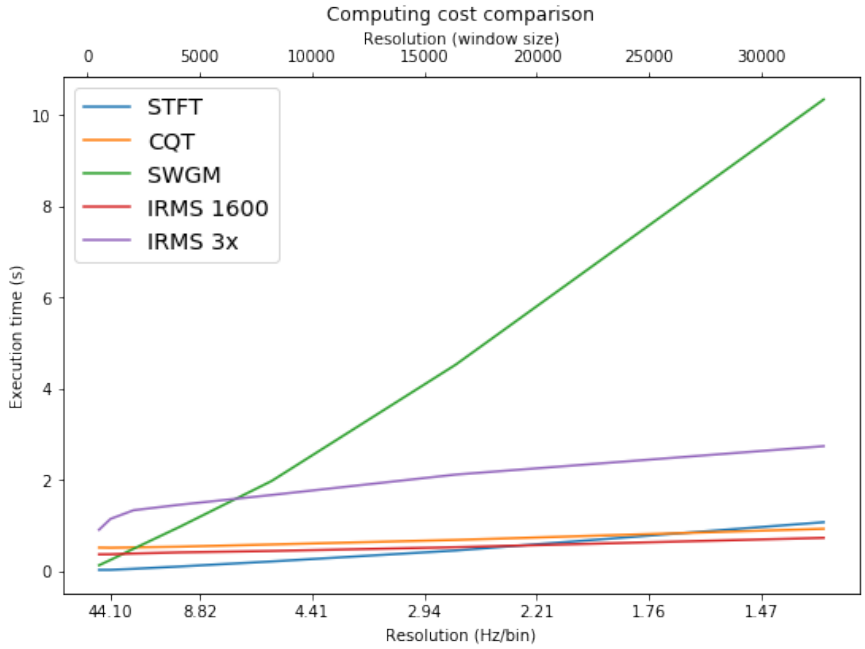


Figure 5.8: *Computational cost curves for selected representations. Using a subregion size of 1600 cents per 1600 ms, the single level IRMS is shown to be cheaper than the CQT for all evaluated resolution points and cheaper than the STFT for resolutions above 2.75 Hz/bin. For resolutions above 4 Hz/bin, the SWGM is not competitive with the other representations shown here. Resolution is given in the corresponding window size for a sampling rate of 44.1 kHz.*

Hz/bin, and demonstrate that the IRMS is successful in producing an adaptive multiresolution representations at a cost comparable with STFT and CQT.

Fig. 5.9 is a comparison of the single-level IRMS with the STFT and CQT. This figure makes it clear that reducing the analysis subregion size increases execution times of the IRMS. This happens because most of the execution time is spent on sub-band processing, as shown in Appendix 6.2. The most computationally expensive parts of this algorithm are the filtering steps, which become more expensive as the frequency band gets smaller: since their requisites (attenuation in stopbands and passbands) are fixed, higher orders are needed for smaller frequency bands.

Taking into account the adaptive aspect of IRMS, it can be said that overall our algorithm compares favorably with the STFT and CQT. The single-level IRMS curves behave approximately linear, with a bigger initial cost (intercept) than the STFT but a smaller gradient. This means that, for frequency resolutions over a certain value (depending on subregion size), the costs of detecting relevant subregions and performing the sub-band processing algorithm are offset by the gains of refining only relevant parts of the spectrogram. This makes our representation an efficient alternative for resolutions above 2.2 Hz/bin. The slope of the CQT sits between the slopes of the STFT and IRMS, and its overall cost for the analyzed resolution range is very similar to the cost of the IRMS with a subregion size of 500 cents per 500 ms.

Fig. 5.10 shows computing times for all tested configurations of the IRMS. Aside from the observations about subregion size, the figure clearly shows that adding a level of refinement is a very costly operation. This means that, with regards to execution time alone, it would seem to be more advantageous to use the single-level configuration with a higher k parameter than adding several levels of refinement (for example, using a single-level that increases the resolution by a factor of 8 instead of three levels where each increases the resolution by a factor of 2). However, there may be other advantages to using the multi-level configurations depending on the subsequent tasks performed on the obtained representation, as will be discussed below.

Fig. 5.11 shows boxplots of execution times for the IRMS tested configurations. The variance for each boxplot is a function of two factors: the refinement percentage, that is defined independently for

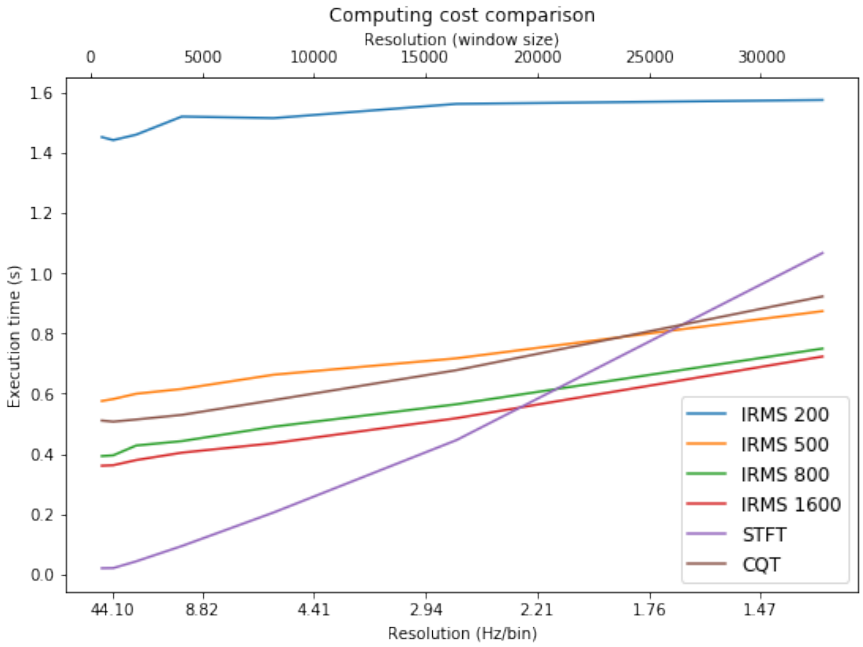


Figure 5.9: *Computational cost curves for the STFT, CQT and every evaluated single-level configuration of the IRMS. Execution time increases as the subregion size of the IRMS decreases, but it still remains comparable to the STFT and CQT for subregions above 500 cents per 500 ms.*

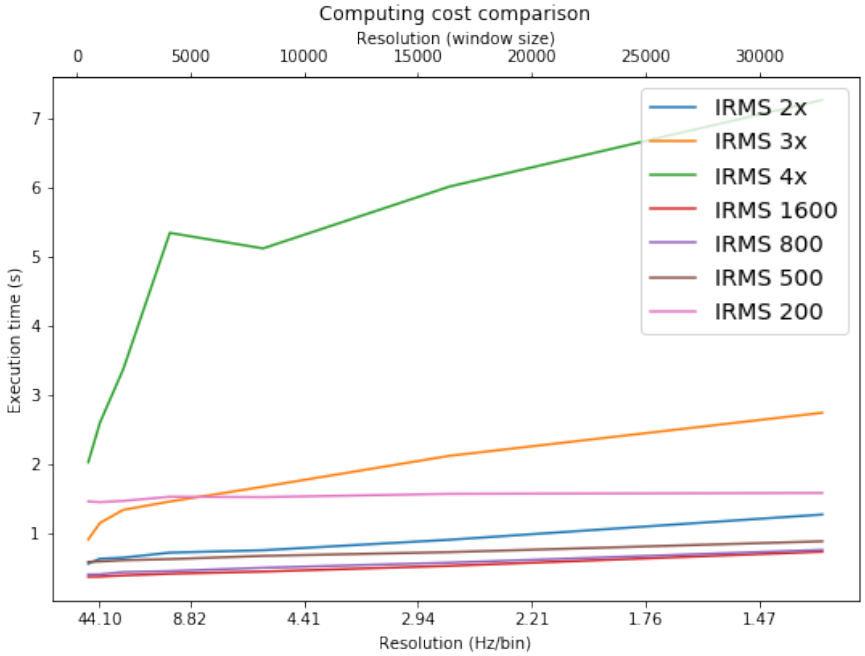


Figure 5.10: *Computational cost curves for all tested configurations of the IRMS. Adding levels of refinement proves to be very costly operation.*

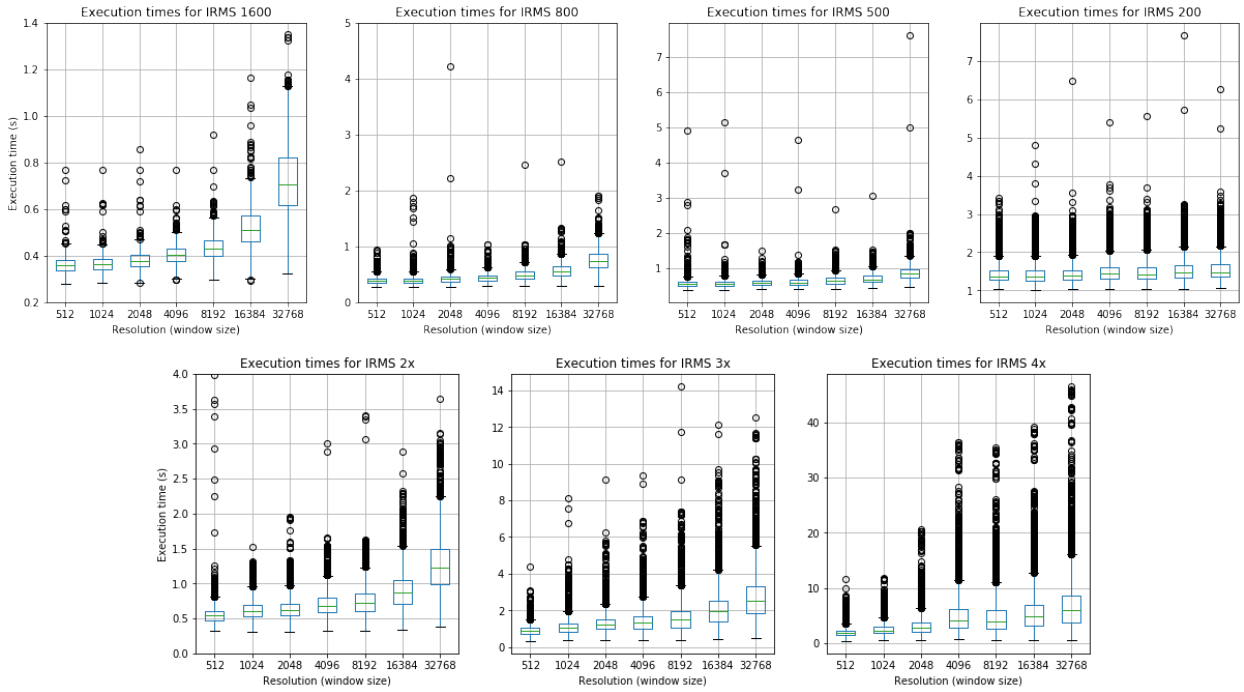


Figure 5.11: *Boxplots of execution times for all tested configurations of the IRMS. Variance for each data point is due to the different refinement percentages and the location of the refined subregions for each file, which change with the analyzed signal and have an influence on the cost of the sub-band processing algorithm. The box extends from the first to the third quartile values of the data, with a line at the median. The whiskers show the range of data, extending no more than* $1.5 * (Q3 - Q1)$ *from the edges of the box, while outliers are plotted as separate dots.*

each processed recording, and the locations of the refined subregions, that depend on signal content. It can be said that the observed variance is expected and even desired in adaptive transforms, since by definition they operate differently when analyzing different signals.

Single-level and multi-level IRMS

We now turn our attention to single-level versus multi-level IRMS representations, and to identifying contexts where the latter would be preferable over the former. Consider, for example, a local MIR detection algorithm (e.g. peak-finding or onset detector); if the target events are all concentrated in the relevant subregions of the last level of refinement of our multiresolution representation, such an algorithm would be applied on a much smaller area of the TFP when additional levels are computed. So, adding additional refinement levels effectively decreases the percentage of the TFP area that would have to be analyzed by event-oriented local MIR detection algorithms. Fig 5.12 shows the total percentage of the TFP area that is part of the last level of refinement. The figure shows that, for the given example, 10% of the spectrogram would be fed to an MIR detection algorithm when using a 4-level IRMS, against 36% when using a single-level IRMS (median values), and 100% when using a conventional spectrogram. This way, the extra cost of a multilevel IRMS could be offset by a speed up in subsequent AMT or MIR algorithms performed on a much smaller subregion of the TFP.

Fig. 5.13 shows boxplots of refinement percentages per subregion size (single-level IRMS) and per refinement level (multi-level IRMS), according to the ground truth values extracted from the corresponding MIDI files. The first plot gives us an idea of how many subregions (as a function of subregion size) of the initial spectrogram's TFP actually contain notes (i.e. fundamental frequencies; recall that the evaluated dataset contains solo piano performances). This figure is useful for the determination of a default value for the refinement percentage of the single-level IRMS for transcription tasks: for subregions between 800×800 and 1600×1600 cents per ms, this percentage

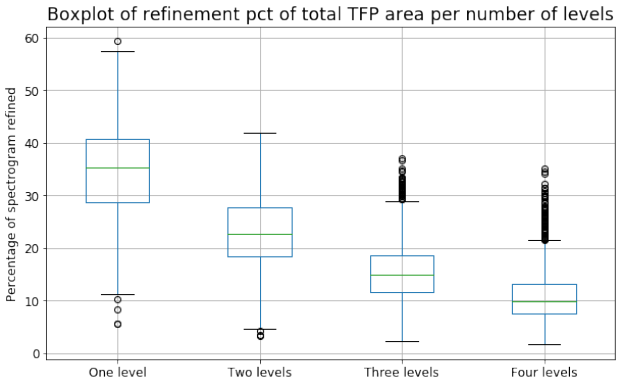


Figure 5.12: *Boxplot showing the percentage of subregions of the initial spectrogram that are processed in the last level of refinement, that is, the percentage of subregions that present the maximum resolution in an IRMS multiresolution representation. These values are calculated for the entire MAESTRO dataset, using MIDI files as reference in order to calculate how many subregions of the TFP contain notes. The "One level" label refers to a single-level IRMS with subregion size of 1600×1600*

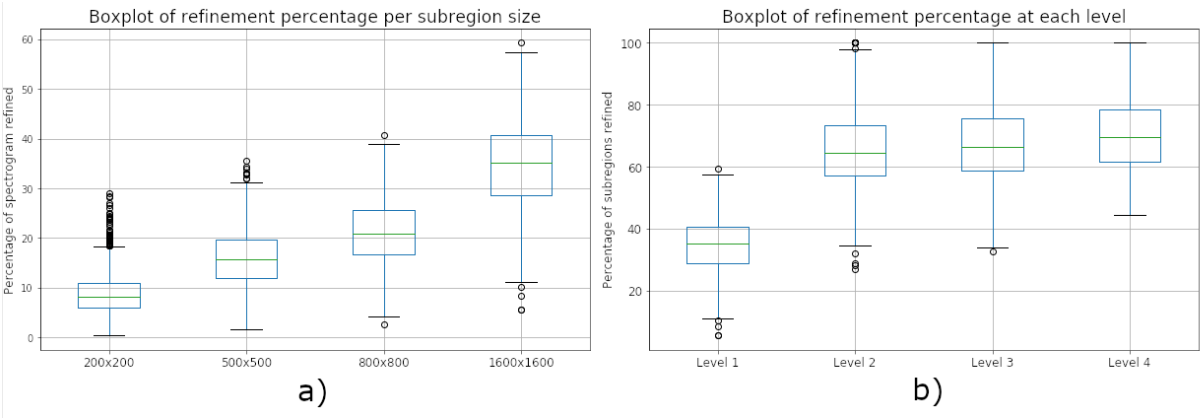


Figure 5.13: ***a)*** *Refinement percentages as a function of subregion size. These values are calculated for the entire MAESTRO dataset, using MIDI files as reference in order to calculate how many subregions of the TFP contain notes using different subregion sizes.* ***b)*** *Refinement percentages at each level of refinement, using subregions of sizes 1600×1600, 800×800, 400×400 and 200×200 on levels 1 to 4. For levels above 1, we calculate the percentage of subregions that contain notes and are inside the regions generated by the previous level of refinement*

lies between 20% and 40% for this dataset. The second plot shows the same analysis for the multi-level IRMS. After the first level of refinement, which drastically reduces the area in the TFP under scrutiny, around 70% of the subregions refined in the preceding level are further refined on each subsequent level. This means that subregions tend to concentrate around actual music events, so more subregions are now considered relevant. Surely this is a by-product of the chosen percentages and subregion sizes (1600×1600 at level 1, 800×800 at level 2, 400×400 at level 3 and 200×200 cents per ms at level 4), but it still gives us an idea of useful refinement percentages for transcription tasks.

Lastly, Fig. 5.14 shows the timing of the IRMS stressed configuration. The maximum refinement percentage found on the MAESTRO dataset for a subregion size of 800 cents per 800 ms was 40.7%, almost two times the mean value of 20.8%. This figure shows that the worst-case scenario results in a considerable rise in execution time when compared with its non-stressed counterpart. Although significant, this increase still leaves the stressed IRMS somewhat competitive with the CQT, and orders of magnitude cheaper than the other evaluated adaptive representations. This result is useful for establishing bounds on IRMS execution time based on refinement percentage, and shows that a high percentage does not invalidate IRMS as an efficient choice for adaptive representations.

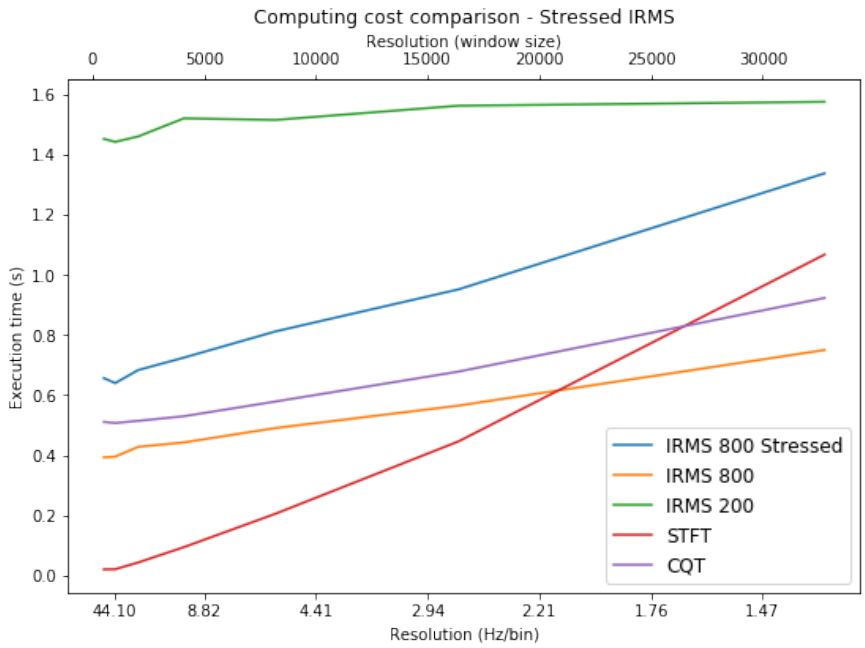


Figure 5.14: *Computational cost curve for the stressed IRMS and other compared representations. Although a high refinement percentage results in a considerable rise in execution time, the IRMS still remains an efficient choice for an adaptive representation.*

5.4.4 Conclusion

In this experiment, we evaluated computational cost curves for the STFT, CQT, Lukin-Todd, SLS, SWGM and several configurations of the IRMS. When compared to other adaptive representations (Lukin-Todd, SLS, SWGM), IRMS performed very well, achieving the smallest execution times and sitting orders of magnitude below SLS and Lukin-Todd. This advantage is especially pronounced for the single-level IRMS. These results validate our initial proposal of an adaptive representation that does not conform to the traditional framework of combining multiple fixed-resolution spectrograms from a precomputed dictionary, and demonstrate its success in providing an efficient alternative to existing adaptive representations.

Furthermore, the single-level IRMS can be faster than the STFT for higher resolutions (finer than 2.2 Hz/bin), where the cost of detecting and refining relevant subregions is offset by the gains of refining a smaller portion of the TFP. This makes the IRMS an efficient alternative when "super resolutions" are desired. This performance can be further improved by speeding up the the filtering stages of the sub-band processing algorithm used for the computation of localized STFTs, e.g., by relaxing filter design criteria (attenuation in stopbands and pass bands).

The experiment suggests a choice of subregion sizes above 500 cents per 500 ms in order to keep execution times competitive with STFT and CQT, and a refinement percentage between 20% and 40% in the first level and around 70% in subsequent levels of the IRMS, at least for the MAESTRO dataset. The cost comparison favors the use of the single-level IRMS against multi-level configurations, although the latter could speed up subsequent AMT and MIR algorithms because of the ultimately smaller area of the maximally refined subregions.

Finally, further experiments are needed to address the application of the IRMS in music transcription tasks. A comparative experiment of IRMS against other representations in tasks such as onset detection and F0 extraction shall reveal the interplay between computational cost and accuracy of the obtained estimates.

Chapter 6

Conclusion

In this work we presented a novel adaptive representation focused on the efficient use of computing power. This proposal is an alternative to the traditional framework of adaptive representations that rely on the combination of a set of precomputed single resolution representations [LT06]. The traditional scheme relies on the calculation of sparsity measurements across the whole set of representations, which is usually cost-intensive and inefficient, since only part of the precomputed data is used in the resulting multiresolution representation. The proposed Iteratively Refined Multiresolution Spectrogram composes an adaptive representation not by combination of precomputed spectrograms, but by successive refinements on top of an initial low-frequency-resolution spectrogram. These refinements are located in the TFP subregions judged to contain relevant music information such as notes, harmonics and expressive elements. The efficiency of the IRMS is two-fold: by using coarser resolutions in areas that do not contain musical events and higher resolutions in musically relevant regions, it reduces memory usage of the final representation; by only computing high resolution representations of the relevant TFP subregions, it reduces execution time while not disregarding any computed high resolution data.

Two main problems were investigated in order to develop the IRMS: the efficient STFT computation of isolated time-frequency regions and the identification of musically relevant regions of a spectrogram. In the former, we reviewed well-known sub-band processing techniques for compression and analysis of music signals. Based on this review, we implemented the "STFT Zoom" tool, a visual interface for the fast computation of high-resolution representations of specific regions of the TFP via sub-band processing. In the latter, we reviewed signal information estimators in the TFP, and proposed a comparative experiment between different features using ground truth values related to MIR tasks. Both Rényi and Shannon entropies showed promising results in their use as music information estimators.

For the integration between detection of relevant subregions and sub-band processing, several empirical evaluations were made leading to optimization choices such as filter design requisites and the signal bank configuration. A data structure was proposed for the representation of a multiresolution spectrogram in a tree-like structure, as well as algorithms for insertion and visualization. A final IRMS implementation was conceived based on the sub-band algorithm part of "STFT Zoom" and a trained model for classification of musically relevant subregions. Its customizable parameters are: number of refinement levels, subregion size, refinement percentage and target frequency resolution for each refinement level.

The final IRMS implementation was compared to other adaptive and non-adaptive representations on a computational cost comparative experiment. Several configurations of our solution were evaluated, leading to observations regarding influence of subregion size, number of refinement levels and refinement percentage on execution time. MIDI files from piano performances were analyzed in order to investigate suitable default refinement percentages in the use of the IRMS for transcription purposes. The IRMS achieved execution times orders of magnitude faster than all other tested adaptive representations, and in some configurations presented a competitive cost with respect to the STFT and CQT. Overall, the results showed that the IRMS is successful in its attempt to offer

an efficient alternative to the traditional framework used for adaptive representations.

6.1 Contributions

In this work, we presented a review of the literature on multiresolution and adaptive representations, sparsity measurements and sub-band processing techniques. An open tool for the detailed visualization of specific regions from a spectrogram called "STFT Zoom"[1] was developed, as a result of the exploration on sub-band processing techniques.

An evaluation of different features for musical information estimation in the TFP was undertaken in a comparative experiment relating them to certain AMT tasks. Apart from the main comparison between features, we highlight as contributions of this study the further validation of the use of Rényi entropy as a measurement of time-frequency information [BFJM01] and the training of a Naive-Bayes model for the detection of musically relevant subregions of a spectrogram using Rényi and Shannon entropies as its features. This study was first presented in the 17th Sound and Music Computing Conference (2020) [FQ20].

A comparison between different time-frequency representations based on execution time is another contribution of this work. We offer as an experiment result the comparative profiling of the STFT, CQT, Lukin-Todd, SLS, SWGM and IRMS representations as a function of frequency resolution. Furthermore, as a part of this experiment, Python implementations of the Lukin-Todd, SLS and SWGM representations were produced. The last two were translated from MATLAB with the original author's supervision, and are part of an ongoing collaboration on a Python package containing the proposed representations in [dC20] and other adaptive representations. An article focused on the IRMS proposal and the computational cost experiment was submitted to ICASSP 2021.

Finally, we highlight as the main contribution of this work the IRMS[2] itself, in both its methodological proposition and its final implementation. The final results of this research project indicate its merits as an efficient adaptive time-frequency representation, and indicate possibilities of further exploration of the proposed framework for adaptive representations as a more efficient option to the traditional one present in the reviewed literature.

6.2 Future work

The IRMS proposition shown here can be expanded in a few different ways. The investigation of different models for the detection of musically relevant subregions of the TFP would be beneficial. In order to truly circumvent the trade-off between frequency and time resolution of the STFT, future work should include research on a detection scheme that is capable of not only detecting these subregions, but whether they should be refined time or frequency-wise. As an initial proposition of discrimination between onset and melodic regions, we suggest the comparison of entropies calculated independently along each axis of a given TFP subregion.

Given the IRMS focus on efficiency, future work shall also include testing of different filter design requisites for the sub-band processing algorithm, with the objective of speeding up its execution time. As noted in 4.2.2, some aspects of the utilized data structure can be improved. We highlight here as possibilities the development of a look-up method that is capable of, given a frequency and time value, returning the most detailed bin of the IRMS data structure that represents this region, and the development of an indexing scheme for the access of the tree-like structure as if it were a simple 2-D matrix, facilitating its employment with ready-to-use MIR algorithms.

Finally, further experiments are needed to address the application of the IRMS in music transcription and other MIR tasks. Beyond the evaluation of its computational efficiency presented here, a comparative experiment of the IRMS against other representations in tasks such as onset

[1] Available on https://github.com/nicolasfigueiredo/stft-zoom
[2] Available on https://github.com/nicolasfigueiredo/IRMS

detection and F0 extraction shall reveal the interplay between computational cost and accuracy of the obtained estimates, and provide additional validation for the IRMS as a relevant time-frequency representation for music signals.

Bibliography

[BB09] S Allen Broughton e Kurt Bryan. Discrete fourier analysis and wavelets. Em *Applications to signal and image processing*. Wiley Online Library, 2009. 30

[BFJM01] Richard G Baraniuk, Patrick Flandrin, Augustus JEM Janssen e Olivier JJ Michel. Measuring time-frequency information content using the rényi entropies. *IEEE Transactions on Information theory*, 47(4):1391–1409, 2001. 8, 9, 40, 44, 56

[BFS+02] Roberto Bresin, Anders Friberg, Johan Sundberg et al. Director musices: The kth performance rules system. *IPSJ Report Music Information Science (MUS)*, 2002(63 (2002-MUS-046)):43–48, 2002. 39

[Bla65] E Donnell Blackham. The physics of the piano. *Scientific american*, 213(6):88–99, 1965. 39

[Bro91] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991. 2, 7

[Bro92] Judith C Brown. Musical fundamental frequency tracking using a pattern recognition method. *The Journal of the Acoustical Society of America*, 92(3):1394–1402, 1992. 2

[BRSS19] Johan Brynolfsson, Isabella Reinhold, Josefin Starkhammar e Maria Sandsten. The matched reassignment applied to echolocation data. Em *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 8236–8240. IEEE, 2019. 8, 9, 18

[CDM15] Tian Cheng, Simon Dixon e Matthias Mauch. Modelling the decay of piano sounds. Em *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 594–598. IEEE, 2015. 39

[Coh92] Leon Cohen. What is a multicomponent signal? Em *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, páginas 113–116. IEEE, 1992. 11

[CS10] ALEXANDRA Craciun e MARTIN Spiertz. Adaptive time frequency resolution for blind source separation. Em *Proceedings of the International Student Conference on Electrical Engineering (POSTER'10)*, volume 10, 2010. 8

[CWF76] Ronald E Crochiere, Susan A Webber e James L Flanagan. Digital coding of speech in sub-bands. *Bell System Technical Journal*, 55(8):1069–1085, 1976. 9, 10

[dC20] Maurício do Vale Madeira da Costa. *NOVEL TIME-FREQUENCY REPRESENTATIONS FOR MUSIC INFORMATION RETRIEVAL*. Tese de Doutorado, Universidade Federal do Rio de Janeiro, 2020. 3, 8, 19, 20, 45, 46, 48, 56

[Dix06] Simon Dixon. Onset detection revisited. Em *Proceedings of the 9th International Conference on Digital Audio Effects*, volume 120, páginas 133–137. Citeseer, 2006. 46

[DLU91] Yves Françcois Dehery, Michel Lever e Pierre Urcun. A musicam source codec for digital audio broadcasting and storage. Em *[Proceedings] ICASSP 91: 1991 International Conference on Acoustics, Speech, and Signal Processing*, páginas 3605–3608. IEEE, 1991. 3, 15

[Eva93] Gianpaolo Evangelista. Pitch-synchronous wavelet representations of speech and music signals. *IEEE transactions on signal processing*, 41(12):3313–3330, 1993. 3

[FQ20] Nicolas Figueiredo e Marcelo Queiroz. Detection of musically relevant regions in multiresolution time-frequency representations evaluated on piano recordings. Em *17th Sound and Music Computing Conference, Torino, Italy*. Zenodo, Junho 2020. 15, 16, 17, 39, 40, 41, 42, 43, 44, 56

[FS99] John Fitch e Wafaa Shabana. A wavelet-based pitch detector for musical signals. *Department of Mathematical Sciences, University of Bath*, 1999. 3

[Gas78] Jack D Gaskill. Linear systems, fourier transforms, and optics. *Linear Systems, Fourier Transforms, and Optics by Jack D. Gaskill New York, NY: John Wiley and Sons, 1978*, 1978. 11

[GH99] Masataka Goto e Satoru Hayamizu. A real-time music scene description system: Detecting melody and bass lines in audio signals. Em *Working Notes of the IJCAI-99 Workshop on Computational Auditory Scene Analysis*, páginas 31–40. Citeseer, 1999. 3

[GSD12] Joachim Ganseman, Paul Scheunders e Simon Dixon. Improving plca-based score-informed source separation with invertible constant-q transforms. Em *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, páginas 2634–2638. IEEE, 2012. 2

[Ham98] Richard Wesley Hamming. *Digital filters*. Courier Corporation, 1998. 10

[HM03] Stephen W Hainsworth e Malcolm D Macleod. Time frequency reassignment: A review and analysis. 2003. 3

[HP02] Hiroshi Harada e Ramjee Prasad. *Simulation and software radio for mobile communications*. Artech House, 2002. 9

[HSR+19] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel e Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. Em *International Conference on Learning Representations*, 2019. 39

[JFB+11] Rajesh Jaiswal, Derry FitzGerald, Dan Barry, Eugene Coyle e Scott Rickard. Clustering nmf basis functions using shifted nmf for monaural sound source separation. Em *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 245–248. IEEE, 2011. 2

[JH17] Nicolas Juillerat e Béat Hirsbrunner. Audio time stretching with an adaptive multiresolution phase vocoder. Em *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, páginas 716–720. IEEE, 2017. 8

[JT07] Florent Jaillet e Bruno Torrésani. Time-frequency jigsaw puzzle: Adaptive multiwindow and multilayered gabor expansions. *International Journal of Wavelets, Multiresolution and Information Processing*, 5(02):293–315, 2007. 3, 8, 13, 18

[K+03] Walt Kester et al. *Mixed-signal and DSP design techniques*. Elsevier, 2003. 9, 10

[KD07] Anssi Klapuri e Manuel Davy. *Signal processing methods for music transcription*. Springer Science & Business Media, 2007. 11

[KG13] Serap Kirbiz e Bilge Günsel. An adaptive time-frequency resolution framework for single channel source separation based on non-negative tensor factorization. Em *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, páginas 905–909. IEEE, 2013. 8

[KRKP+16] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla e Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. Em F. Loizides e B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, páginas 87 – 90. IOS Press, 2016. 63

[LBR11] Marco Liuni, Peter Balazs e Axel Röbel. Sound analysis and synthesis adaptive in time and two frequency bands. *arXiv preprint arXiv:1109.6651*, 2011. 9

[LG91] Didier Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–59, 1991. 3, 15

[LRM+13] Marco Liuni, Axel Robel, Ewa Matusiak, Marco Romito e Xavier Rodet. Automatic adaptation of the time-frequency resolution for sound analysis and re-synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5):959–970, 2013. 3, 8, 9, 18

[LT06] Alexey Lukin e Jeremy Todd. Adaptive time-frequency resolution for analysis and processing of audio. Em *Audio Engineering Society Convention 120*. Audio Engineering Society, 2006. 3, 7, 8, 20, 45, 48, 55

[MdVB17] da Costa Maurício do VM e Luiz WP Biscainho. Combining time-frequency representations for music information retrieval. Em *15o Congresso de Engenharia de Áudio da AES-Brasil, Florianópolis, Brazil*, páginas 12–18, 2017. 19, 45

[MMB+19] Brian McFee, Matt McVicar, Stefan Balke, Vincent Lostanlen, Carl Thomé, Colin Raffel, Dana Lee, Kyungyun Lee, Oriol Nieto, Frank Zalkow, Dan Ellis, Eric Battenberg, Ryuichi Yamamoto, Josh Moore, Ziyao Wei, Rachel Bittner, Keunwoo Choi, nullmightybofo, Pius Friesch, Fabian-Robert Stöter, Thassilo, Matt Vollrath, Siddhartha Kumar Golu, nehz, Simon Waloschek, Seth, Rimvydas Naktinis, Douglas Repetto, Curtis "Fjord" Hawthorne e CJ Carr. librosa/librosa: 0.6.3, Fevereiro 2019. 25, 40, 48

[Moo95] Brian CJ Moore. *Hearing*. Academic Press, 1995. 1

[NMG+10] Juhan Nam, Gautham J Mysore, Joachim Ganseman, Kyogu Lee e Jonathan S Abel. A super-resolution spectrogram using coupled plca. Em *Eleventh Annual Conference of the International Speech Communication Association*, 2010. 3

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot e E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 43

[Rot83] Joseph Rothweiler. Polyphase quadrature filters–a new subband coding technique. Em *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 8, páginas 1280–1283. IEEE, 1983. 3

[SGER14] Justin Salamon, Emilia Gómez, Daniel PW Ellis e Gaël Richard. Melody extraction from polyphonic music signals: Approaches, applications, and challenges. *IEEE Signal Processing Magazine*, 31(2):118–134, 2014. 18

[SK10] Christian Schörkhuber e Anssi Klapuri. Constant-q transform toolbox for music processing. Em *7th Sound and Music Computing Conference, Barcelona, Spain*, páginas 3–64, 2010. 3, 9, 15, 19, 45

[SKHD14] Christian Schörkhuber, Anssi Klapuri, Nicki Holighaus e Monika Dörfler. A matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution. Em *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014. 2

[SKS13] Christian Schörkhuber, Anssi Klapuri e Alois Sontacchi. Audio pitch shifting using the constant-q transform. *Journal of the Audio Engineering Society*, 61(7/8):562–572, 2013. 2

[TC79] Jose Tribolet e Ronald Crochiere. Frequency domain coding of speech. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(5):512–530, 1979. 3, 9, 10

[VSW91] Rodney G Vaughan, Neil L Scott e D Rod White. The theory of bandpass sampling. *IEEE Transactions on signal processing*, 39(9):1973–1984, 1991. 3, 9

[Wal92] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992. 3, 15

[WBH91] William J Williams, Mark L Brown e Alfred O Hero. Uncertainty, information, and time-frequency distributions. Em *Advanced Signal Processing Algorithms, Architectures, and Implementations II*, volume 1566, páginas 144–157. International Society for Optics and Photonics, 1991. 8, 9

[ZN77] Rainer Zelinski e Peter Noll. Adaptive transform coding of speech signals. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(4):299–309, 1977. 3, 18

IRMS Code Profiling

This appendix presents execution time profiling for the implemented code of the IRMS representation. We intend to give a top-down view of the algorithm stages, while identifying the most cost-intensive parts of the code. For the sake of clarity, we present the algorithm of the single-level IRMS only. The multi-level profiling would follow the same structural observations made here, with some difference of total execution time for each refinement level. The presented figures were generated using the `line_profiler`[3] Python package as a Jupyter Notebook [KRKP+16] extension.

Figure 1 shows the entry point function for the IRMS. The example given is a computation of a single-level IRMS with subregion size equal to 800 cents per 800 ms, refinement percentage of 50% and a k factor of 5. Total execution time lasted 1.0049 seconds. This time was partitioned like so:

- **2.7%** was spent on the initial spectrogram's computation;
- **2.1%** was spent on the initialization of the custom IRMS data structure;
- **8.0%** was spent on the detection (feature extraction + model prediction) of interesting subregions;
- **39.3%** was spent initializing the signal bank of the `stft_zoom` script;
- **40.6%** was spent performing localized STFTs with the implemented sub-band processing algorithm;
- **4.7%** was spent inserting these localized STFTs into the IRMS data structure.

Now, we take a closer look at some of these stages in the sequential order of the algorithm, starting with the detection stage shown in Fig. 2. This stage's execution time is taken up almost entirely (99.5%) by feature extraction, while probability prediction of the Naive-Bayes model takes up only 0.4% of this time.

Fig. 3 shows a function call of the `stft_zoom()` function, that performs the sub-band processing algorithm for localized STFT computation. In this case, the processed subregion is found in the signal bank, so most of the sub-band processing algorithm is bypassed: we only need to find the corresponding signal in the signal bank, slice it and perform the localized STFT. 80.8% of its total time of 0.00181 seconds is spent on the slicing and STFT operations, while 13.3% is spent on an utility function that composes the x-axis and y-axis values of the STFT matrix.

Fig. 4 shows a function call of the `stft_zoom()` function for when a subregion is not found on the signal bank. In this case, the function took 0.0107 seconds to run, from which 87.8% is taken up by the filtering and modulation steps, 0.2% is taken up by subsampling and 9.1% is taken up by the STFT computation of the sub-band processed signal. Fig. 5 takes a closer look at the filtering and modulation steps. 44.8% of its execution time is spent on band-pass filtering, while 54.9% is spent on ring modulation followed by the final low-pass filter stage.

In conclusion, the IRMS execution time is mostly a function of its sub-band processing algorithm, which takes up 79.9% of its total execution time (39.3% setting up the signal bank and 40.6% performing additional sub-band processing computations). Most (about 87.8%) of this 79.9%, in

[3] https://github.com/pyutils/line_profiler

```
Timer unit: 1e-06 s

Total time: 1.0049 s
File: <ipython-input-19-eff08cefd601>
Function: ers_singlelevel at line 14

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    14                                           def ers_singlelevel(y, res, kernel, model, pct, sr=44100, n_fft=512, hop_size=512):
    15         1      26847.0  26847.0      2.7      spec = np.abs(librosa.stft(y, n_fft=n_fft, hop_length=hop_size))
    16         1         13.0     13.0      0.0      time_span = [0,len(y)/sr]
    17         1        511.0    511.0      0.1      x_axis, y_axis = stft_zoom.get_axes_values(sr, 0, time_span, spec.shape)
    18         1      21065.0  21065.0      2.1      base_spec = SingleResSpectrogram(spec, x_axis, y_axis)
    19         1         45.0     45.0      0.0      multires_spec = MultiResSpectrogram(base_spec)
    20         1      80165.0  80165.0      8.0      indices, original_shape = detect_musical_regions.detect_musical_regions(model, spec, kern
    21         1      16692.0  16692.0      1.7      to_be_refined = detect_musical_regions.musical_regions_to_ranges(indices, original_shape,
    22
    23         1     394490.0 394490.0     39.3      stft_zoom.set_signal_bank(y,kernel)
    24
    25       223        310.0      1.4      0.0      for subregion in to_be_refined:
    26       222        201.0      0.9      0.0          freq_range = subregion[0]
    27       222        177.0      0.8      0.0          time_range = subregion[1]
    28       222     407648.0   1836.3     40.6          spec_zoom, x_axis, y_axis, new_sr, window_size, hop_size = stft_zoom.stft_zoom(y, fre
    29       222       9540.0     43.0      0.9          refined_subspec = SingleResSpectrogram(spec_zoom, x_axis, y_axis)
    30       222      47197.0    212.6      4.7          multires_spec.insert_zoom(multires_spec.base_spec, refined_subspec, zoom_level=1)
    31
    32         1          0.0      0.0      0.0      return multires_spec
```

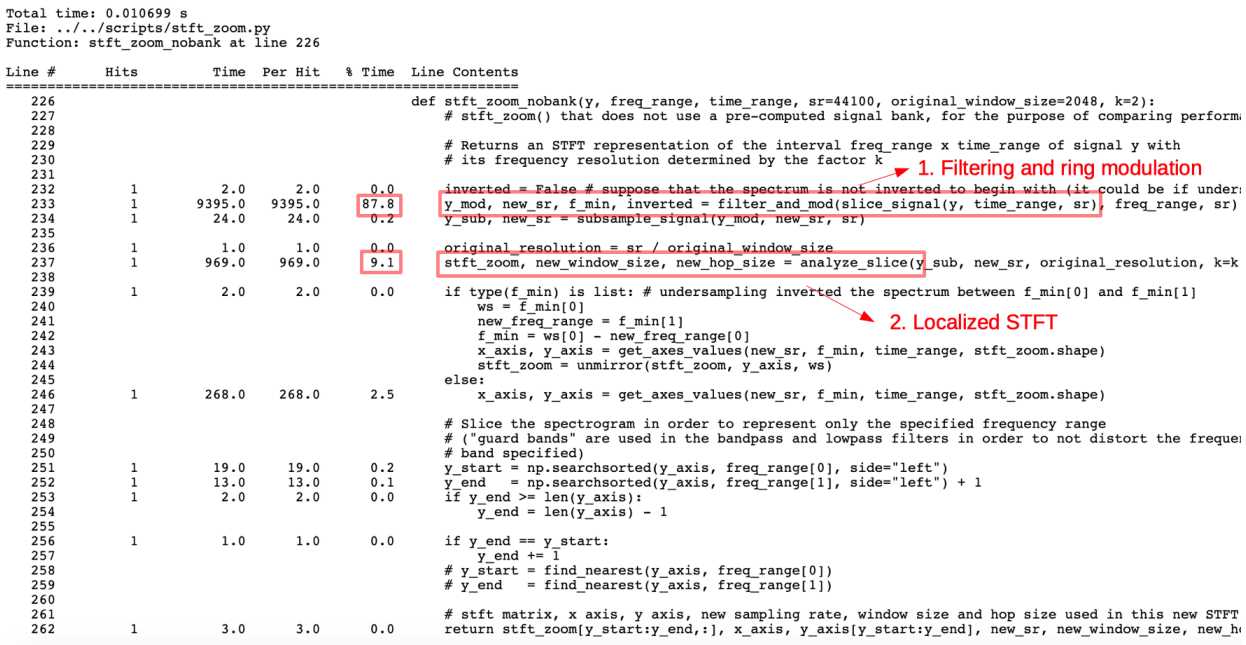

Figure 1: *Cost profiling of the entry point function for the IRMS, with annotations showing the detection, signal bank and sub-band processing code lines*

```
Timer unit: 1e-06 s

Total time: 0.109634 s
File: ../../scripts/detect_musical_regions.py
Function: detect_musical_regions at line 12

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    12                                           def detect_musical_regions(model, spectrogram, mode='threshold', pct_or_threshold=0.75, kernel=[800, 800
    13                                               # Takes a spectrogram and detects its musically interesting subregions, according to a NB model give
    14                                               # and a threshold of probability or percentile.
    15                                               # The returned list are the flattened indices that represent interesting subregions in the feature m
    16                                               # should be converted later
    17
    18                                               # Compute features
    19         1     109140.0 109140.0     99.5      shannon, renyi = mappings.extract_features(spectrogram, kernel, n_fft=n_fft, hop_size=hop_size, sr=s
    20         1         16.0     16.0      0.0      X = np.array([shannon.flatten(), renyi.flatten()]).T
    21
    22                                               # Predict probability
    23         1        440.0    440.0      0.4      predicted_probs = model.predict_proba(X)
    24
    25                                               # Filter results according to threshold and return a sorted list of indices
    26         1          1.0      1.0      0.0      if mode == 'threshold':
    27                                                   sorted_probs = np.sort(predicted_probs[:,1])[::-1]
    28                                                   idx = sorted_probs.size - np.searchsorted(sorted_probs[::-1], pct_or_threshold, side = "right")
    29         1          1.0      1.0      0.0      elif mode == 'pct':
    30         1          7.0      7.0      0.0          idx = int(len(renyi.flatten()) * (pct_or_threshold/100))
    31         1         29.0     29.0      0.0      return np.argsort(predicted_probs[:,1])[::-1][:idx], shannon.shape  # shape of feature map is used l
```


Figure 2: *Cost profiling for the detection phase of the IRMS, highlighting the feature extraction and model prediction parts of the code*

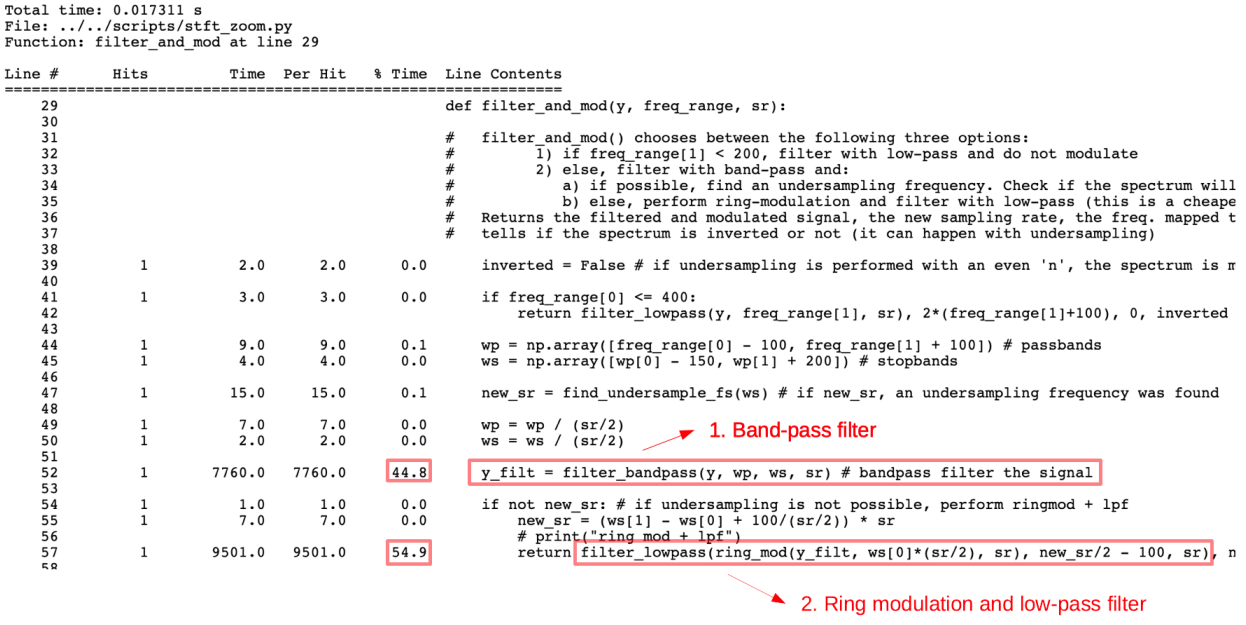
```
Total time: 0.00181 s
File: ../../scripts/stft_zoom.py
Function: stft_zoom at line 184

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   184                                           def stft_zoom(y, freq_range, time_range, sr=44100, original_window_size=2048, k=2):
   185                                               # Returns an STFT representation of the interval freq_range x time_range of signal y with
   186                                               # its frequency resolution determined by the factor k
   187                                               global y_bank, bank_idx
   188
   189         1          1.0      1.0      0.1      inverted = False # suppose that the spectrum is not inverted to begin with (it could be if undersampling
   190
   191         1          7.0      7.0      0.4      if freq_range[0] > 100 and freq_range[0] < 3000:
   192                                                   # print(freq_range, "used signal bank")
   193         1         40.0     40.0      2.2          y_mod, new_sr, f_min, inverted = y_bank[get_bank_idx(freq_range)]
   194         1          6.0      6.0      0.3          y_mod = slice_signal(y_mod, time_range, sr)
   195                                               else:
   196                                                   # print(freq_range, "did not use signal bank")
   197                                                   y_mod, new_sr, f_min, inverted = filter_and_mod(slice_signal(y, time_range, sr), freq_range, sr)
   198         1         11.0     11.0      0.6      y_sub, new_sr = subsample_signal(y_mod, new_sr, sr)
   199
   200         1          1.0      1.0      0.1      original_resolution = sr / original_window_size
   201         1       1462.0   1462.0     80.8      stft_zoom, new_window_size, new_hop_size = analyze_slice(y_sub, new_sr, original_resolution, k=k)
   202
   203         1          3.0      3.0      0.2      if type(f_min) is list: # undersampling inverted the spectrum between f_min[0] and f_min[1]
   204                                                   ws = f_min[0]
   205                                                   new_freq_range = f_min[1]
   206                                                   f_min = ws[0] - new_freq_range[0]
   207                                                   x_axis, y_axis = get_axes_values(new_sr, f_min, time_range, stft_zoom.shape)
   208                                                   stft_zoom = unmirror(stft_zoom, y_axis, ws)
   209                                               else:
   210         1        240.0    240.0     13.3          x_axis, y_axis = get_axes_values(new_sr, f_min, time_range, stft_zoom.shape)
   211
   212                                               # Slice the spectrogram in order to represent only the specified frequency range
   213                                               # ("guard bands" are used in the bandpass and lowpass filters in order to not distort the frequency
   214                                               # band specified)
   215         1         18.0     18.0      1.0      y_start = np.searchsorted(y_axis, freq_range[0], side="left")
   216         1         13.0     13.0      0.7      y_end   = np.searchsorted(y_axis, freq_range[1], side="left") + 1
   217         1          3.0      3.0      0.2      if y_end >= len(y_axis):
   218                                                   y_end = len(y_axis) - 1
   219
   220         1          1.0      1.0      0.1      if y_end == y_start:
   221                                                   y_end += 1
   222
   223                                               # stft matrix, x axis, y axis, new sampling rate, window size and hop size used in this new STFT
   224         1          4.0      4.0      0.2      return stft_zoom[y_start:y_end,:], x_axis, y_axis[y_start:y_end], new_sr, new_window_size, new_hop_size
```


Figure 3: *Code profiling for the sub-band processing stage of the IRMS, in the case when a subregion is found on the stft_ zoom's signal bank.*



```
Total time: 0.010699 s
File: ../../scripts/stft_zoom.py
Function: stft_zoom_nobank at line 226

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   226                                           def stft_zoom_nobank(y, freq_range, time_range, sr=44100, original_window_size=2048, k=2):
   227                                               # stft_zoom() that does not use a pre-computed signal bank, for the purpose of comparing perform
   228
   229                                               # Returns an STFT representation of the interval freq_range x time_range of signal y with
   230                                               # its frequency resolution determined by the factor k
   231                                                                                                                    1. Filtering and ring modulation
   232         1          2.0      2.0      0.0      inverted = False # suppose that the spectrum is not inverted to begin with (it could be if under
   233         1       9395.0   9395.0     87.8      y_mod, new_sr, f_min, inverted = filter_and_mod(slice_signal(y, time_range, sr), freq_range, sr)
   234         1         24.0     24.0      0.2      y_sub, new_sr = subsample_signal(y_mod, new_sr, sr)
   235
   236         1          1.0      1.0      0.0      original_resolution = sr / original_window_size
   237         1        969.0    969.0      9.1      stft_zoom, new_window_size, new_hop_size = analyze_slice(y_sub, new_sr, original_resolution, k=k
   238
   239         1          2.0      2.0      0.0      if type(f_min) is list: # undersampling inverted the spectrum between f_min[0] and f_min[1]
   240                                                   ws = f_min[0]
   241                                                   new_freq_range = f_min[1]                                        2. Localized STFT
   242                                                   f_min = ws[0] - new_freq_range[0]
   243                                                   x_axis, y_axis = get_axes_values(new_sr, f_min, time_range, stft_zoom.shape)
   244                                                   stft_zoom = unmirror(stft_zoom, y_axis, ws)
   245                                               else:
   246         1        268.0    268.0      2.5          x_axis, y_axis = get_axes_values(new_sr, f_min, time_range, stft_zoom.shape)
   247
   248                                               # Slice the spectrogram in order to represent only the specified frequency range
   249                                               # ("guard bands" are used in the bandpass and lowpass filters in order to not distort the freque
   250                                               # band specified)
   251         1         19.0     19.0      0.2      y_start = np.searchsorted(y_axis, freq_range[0], side="left")
   252         1         13.0     13.0      0.1      y_end   = np.searchsorted(y_axis, freq_range[1], side="left") + 1
   253         1          2.0      2.0      0.0      if y_end >= len(y_axis):
   254                                                   y_end = len(y_axis) - 1
   255
   256         1          1.0      1.0      0.0      if y_end == y_start:
   257                                                   y_end += 1
   258                                               # y_start = find_nearest(y_axis, freq_range[0])
   259                                               # y_end   = find_nearest(y_axis, freq_range[1])
   260
   261                                               # stft matrix, x axis, y axis, new sampling rate, window size and hop size used in this new STFT
   262         1          3.0      3.0      0.0      return stft_zoom[y_start:y_end,:], x_axis, y_axis[y_start:y_end], new_sr, new_window_size, new_h
```

Figure 4: *Code profiling for the sub-band processing stage of the IRMS, in the case when a subregion is not found on the stft_ zoom's signal bank.*



```
Total time: 0.017311 s
File: ../../scripts/stft_zoom.py
Function: filter_and_mod at line 29

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    29                                           def filter_and_mod(y, freq_range, sr):
    30
    31                                               #   filter_and_mod() chooses between the following three options:
    32                                               #         1) if freq_range[1] < 200, filter with low-pass and do not modulate
    33                                               #         2) else, filter with band-pass and:
    34                                               #            a) if possible, find an undersampling frequency. Check if the spectrum will
    35                                               #            b) else, perform ring-modulation and filter with low-pass (this is a cheape
    36                                               #   Returns the filtered and modulated signal, the new sampling rate, the freq. mapped t
    37                                               #   tells if the spectrum is inverted or not (it can happen with undersampling)
    38
    39         1          2.0      2.0      0.0      inverted = False # if undersampling is performed with an even 'n', the spectrum is n
    40
    41         1          3.0      3.0      0.0      if freq_range[0] <= 400:
    42                                                   return filter_lowpass(y, freq_range[1], sr), 2*(freq_range[1]+100), 0, inverted
    43
    44         1          9.0      9.0      0.1      wp = np.array([freq_range[0] - 100, freq_range[1] + 100]) # passbands
    45         1          4.0      4.0      0.0      ws = np.array([wp[0] - 150, wp[1] + 200]) # stopbands
    46
    47         1         15.0     15.0      0.1      new_sr = find_undersample_fs(ws) # if new_sr, an undersampling frequency was found
    48
    49         1          7.0      7.0      0.0      wp = wp / (sr/2)                                  1. Band-pass filter
    50         1          2.0      2.0      0.0      ws = ws / (sr/2)
    51
    52         1       7760.0   7760.0     44.8      y_filt = filter_bandpass(y, wp, ws, sr) # bandpass filter the signal
    53
    54         1          1.0      1.0      0.0      if not new_sr: # if undersampling is not possible, perform ringmod + lpf
    55         1          7.0      7.0      0.0          new_sr = (ws[1] - ws[0] + 100/(sr/2)) * sr
    56                                                   # print("ring_mod + lpf")
    57         1       9501.0   9501.0     54.9          return filter_lowpass(ring_mod(y_filt, ws[0]*(sr/2), sr), new_sr/2 - 100, sr), n
    58
                                                                                                                    2. Ring modulation and low-pass filter
```

Figure 5: *Code profiling for the filtering and ring modulation phases of the sub-band algorithm, higlighting the band-pass filter and low-pass filter sections of the code.*

turn, is spent on the filtering stages of the sub-band algorithm, whether low-pass or band-pass. This means that efforts to speed up the algorithm should tackle filtering first, whether by relaxing some of the filter design constraints or by employing different filter architectures. Secondly, efforts on speeding up the detection phase should be focused on the faster extraction of features from the initial rough spectrogram of the IRMS.