Chapter 1

Audio Source Separation in a Musical Context

One of the great advances of the 20th and 21st centuries has been the introduction and refinement of audio recording and the audio editing techniques recording allows. Remixing, editing and remastering in the studio has enabled the creation of new genres of music (e.g., *musique concrete*, modern hip hop). When instruments are recorded in isolation, modern editing and mixing tools allow correction of small errors in music recordings without requiring a group to re-record an entire passage (e.g., a single missed note in the flute part). This also allows rebalancing of levels between musicians without re-recording (e.g., increasing the volume of the flute compared to the trumpet), and application of audio effects to individual instruments (e.g., adding reverberation to the vocals, but not the bass).

Many of these editing techniques require (nearly) isolated instrumental recordings. One cannot edit out a missed note in the flute, if that note is also recorded on the adjacent microphone for the violin. Unfortunately, there are many recording situations (e.g., a stereo recording of a 10-piece ensemble) where there are many more instruments than there are microphones. Thus, instruments are not recorded in isolation. Also, many legacy recordings are only available in the final stereo (two-channel) or mono (one-channel) mixture and the original pre-mixed tracks are not available. All of these situations make many editing or remixing tasks difficult or impossible.

Audio source separation is the process of extracting individual sound sources (e.g., a single flute) from a mixture of sounds (e.g., a recording of a concert band using a single microphone). Effective source separation would allow application of editing and remixing techniques to existing recordings with multiple instruments on a single track.

There have been many source separation approaches developed for general audio signals. Examples of general source separation algorithms include Independent Component Analysis (ICA) [1], Non-negative Matrix Factorization (NMF) [2], Non-negative Tensor Factorization (NTF) [3], Probabilistic Latent Component Analysis (PLCA) [4], and Robust Principal Component Analysis (RPCA) [5]. While all of these have been applied to music, most do not leverage any particular features of music to aid in the separation process.

One element of a music scene which is highly salient is the repeating structure found in the music. Schenker asserted that repetition is what gives rise to the concept of the motive [6]. Ruwet used repetition as a criterion for dividing music into small parts, revealing the syntax of the musical piece [7]. Ockelford argued that repetition and imitation is what brings order to music [8].

Computational audio researchers have found repetitive elements in music audio useful for many purposes. Common applications include music summarization. [9, 10, 11], audio segmentation [12], beat estimation [13], finding drum patterns in the audio [14], and structural analysis [15]. For a thorough review on music structure analysis, the reader is referred to [11], [16] and [17].

The idea that repetition can be used for source separation is supported by recent findings in psychoacoustics. McDermott et al. established that the human auditory system is able to segregate individual sources by identifying them as repeating patterns embedded in the acoustic input, without requiring prior knowledge of the source properties [18]. Given this, could repetition in music be used as a basis for automatically separating the audio into a repeating background (e.g. a salsa montuno) and a varied foreground (the lead 'salsero' sala singer)?

Another salient feature of a musical scene is melody. Bregman [19] shows that humans often link together distinct sound elements (e.g. notes) in time to produce perceptually salient entities called auditory streams. These streams often correlate strongly with melodies. Many trained musicians are able to segregate several melodies into independent elements they can attend to. Can a machine do the same? Often, a musical score can aid the trained musician to perform this task better. Can a score provide similar aid to an auditory streaming algorithm?

In this chapter we will focus on a pair of source separation approaches designed to work with music audio. The first seeks the repeated elements in the musical scene and separates th repeating from the non-repeating. The second looks for melodic elements, pitch tracking and streaming the audio into separate elements. This second approach is then informed by a musical score to improve performance.

1.1 REPET

In this work, we begin with the observation that passages in many kinds of folk and pop music can be understood as a background component that is generally repeating in time, with a superimposed foreground component that is generally variable in time (e.g., a repeating accompaniment superimposed with varying vocals or a solo instrument). On this basis the *REpeating Pattern Extraction Technique (REPET)* was proposed. REPET is an intuitive approach for separating the repeating background from the non-repeating foreground in an audio mixture. The basic idea is to identify repeating elements in the mixture by measuring self-similarity along time, derive repeating models by averaging the repeating elements over their repetition rates, and extract the repeating structure by comparing the repeating models to the mixture.

A number of experiments have shown that REPET can be effectively applied for separating pop songs into their music accompaniment and singing voice. Unlike other approaches for source separation, REPET does not depend on special parametrizations, does not rely on complex frameworks, and does not require external information. Because it is only based on repetition, it has the advantage of being simple, fast, blind, and therefore completely and easily automatable. More information about REPET, including source code, audio examples, and related publications can be found at `http://music.eecs.northwestern.edu/research.php?project=repet`.

1.1.1 Original REPET

The original REPET algorithm was designed to separate the repeating background from the non-repeating foreground in an audio mixture (e.g., the music accompaniment from the singing voice in a pop song) by identifying a period and modeling a segment for the periodically repeating patterns [20, 21].

The method can be summarized in three stages (see Figure 1.1): (1) identification of a repeating period; (2) modeling of a repeating segment; and (3) extraction of the repeating structure.

Repeating Period Identification

In the first stage, the time-domain signal (top left of Figure 1.1) is transformed into a time-frequency representation known as the spectrogram by using the Short-Time Fourier Transform (STFT). A spectrogram represents sound as a two-dimensional structure, where the vertical axis shows frequency from low (at the bottom) to high. The horizontal axis shows time, from left to right. The spectrograms in Figure

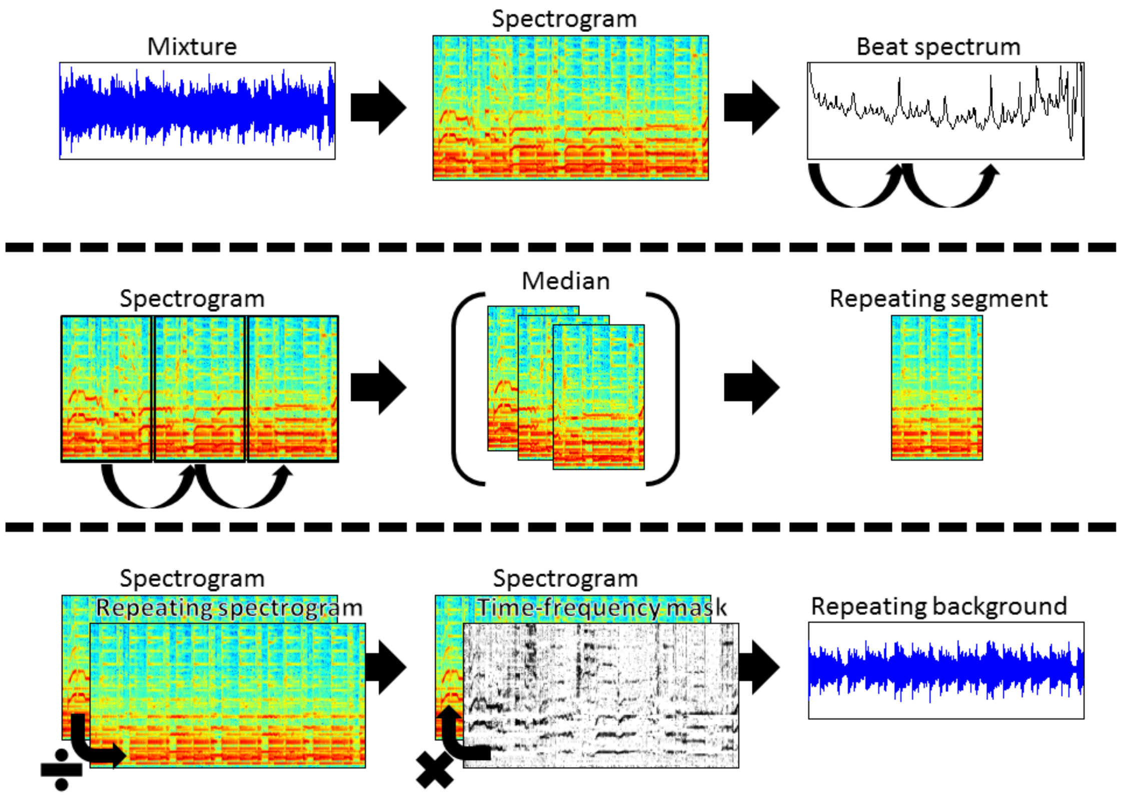

Figure 1.1: Overview of the original REPET: (1) computation of the beat spectrum and identification of a repeating period (top row); (2) filtering of the spectrogram and modeling of a repeating segment (middle row); (3) derivation of the time-frequency mask and extraction of the repeating background (bottom row).

1.1 use red to indicate high energy and blue to show low energy.

The autocorrelation of each frequency channel is then computed. An autocorrelation measures the similarity of a signal with a delayed version of itself given different delays. The peak of the autocorrelation function indicates the period at which the sound repeats. To identify periodicity in the mixture, the beat spectrum [13] is derived from the spectrogram by computing the autocorrelation over time for every frequency channel and averaging the autocorrelations over the frequency channels. This is shown in the upper right of Figure 1.1. Here, a peak is a candidate period at which the audio may repeat.

If periodically repeating patterns are present in the mixture, the beat spectrum forms peaks that are periodically repeating at different period rates, unveiling the underlying periodically repeating structure of the mixture, as shown in Figure 1.1. A best-fit repeating period can then be identified from the beat spectrum, manually or by using an automatic period finder [20, 21].

Repeating Segment Modeling

In the second stage, the repeating period is used to find the period of the underlying repeating structure in the recording. This typically correlates to a pattern a few seconds long, such as a four-chord repeating riff in a folk song. A model of what is repeating in the music is built by segmenting the audio at the points where the pattern repeats (middle panel of Figure 1.1). The median value of the sound across all repetitions is then used to model the canonical repeating sound, as shown in Figure 1.1.

This approach assumes the non-repeating foreground (e.g., vocals or an instrumental soloist) has a sparse and varied time-frequency representation compared with the time-frequency representation of the repeating background. Therefore, time-frequency bins with small deviations at their repetition rate would most likely represent repeating elements and would be captured by the median model. On the other hand, time-frequency bins with large deviations at their repetition rate would most likely be non-repeating elements (i.e., the non-repeating musical foreground) and would be removed by the median model.

Repeating Structure Extraction

In the third stage, the repeating segment is used to derive a repeating spectrogram by taking, for every time-frequency bin, the minimum between the repeating model and the mixture spectrogram at period rate. This assumes the mixture spectrogram is the sum of a non-negative repeating spectrogram and a non-negative non-repeating spectrogram. Thus, the mixture at any given time and frequency is assumed to always be as loud or louder than the individual mixture components (i.e., the repeating components are assumed not to cancel out the non-repeating components).

The repeating spectrogram is then used to derive a time-frequency mask by dividing, for every time-frequency bin, the repeating spectrogram by the mixture spectrogram, as shown in Figure 1.1. The rationale is that, time-frequency bins that are likely to repeat at their repetition rate in the mixture spectrogram would have values near one in the time-frequency mask and would be weighted toward the repeating background. On the other hand, time-frequency bins that are not likely to repeat at their repetition rate in the mixture spectrogram would have values near zeros in the time-frequency mask and would be weighted toward the non-repeating foreground.

The repeating background can then be obtained by multiplying, for every time-frequency bin, the time-frequency mask with the mixture. The non-repeating foreground can be obtained by simply subtracting the repeating background from the mixture.

Experiments showed that REPET can be effectively applied for separating pop/rock song clips into their accompaniment music and singing voice [20, 21]. They also showed that REPET can be combined with other methods to improve background/foreground separation; for example, it can be used as a preprocessor to pitch detection algorithms to improve melody extraction [21], or as a post-processor to a singing voice separation algorithm to improve music/voice separation [22]. Experiments further showed that REPET can be effectively applied for separating full-track real-world songs into their accompaniment music and singing voice, by simply applying the method along time via a sliding window [21]. There is, however, a trade-off for the window size in REPET: if the window is too long, the repetitions will not be sufficiently stable; if the window is too short, there will not be sufficient repetitions [21].

1.1.2 Adaptive REPET

Adaptive REPET is an extension of the original REPET and is designed to handle a varying periodic background, i.e., when the repeating period and/or the repeating patterns change over time (e.g., the succession of two homogenous sections in a song, such as a verse followed by the chorus), without the need for segmenting or windowing the audio [23].

As with the original REPET, the method can be summarized in three stages (see Figure 1.2): (1) identification of the repeating periods; (2) modeling of a repeating spectrogram; and (3) extraction of the repeating structure.

Identification of Repeating Periods

In the first stage, the signal is transformed into a spectrogram. To identify local periodicities in the mixture, the beat spectrogram [13] is derived from the spectrogram by computing a beat spectrum for every time frame by sliding a window along time. In other words, each column in the beat spectrogram represents a beat spectrum at a given time.

If periodically repeating patterns are present in the mixture, the beat spectrogram forms horizontal lines that are periodically repeating vertically, corresponding to the succession of peaks in

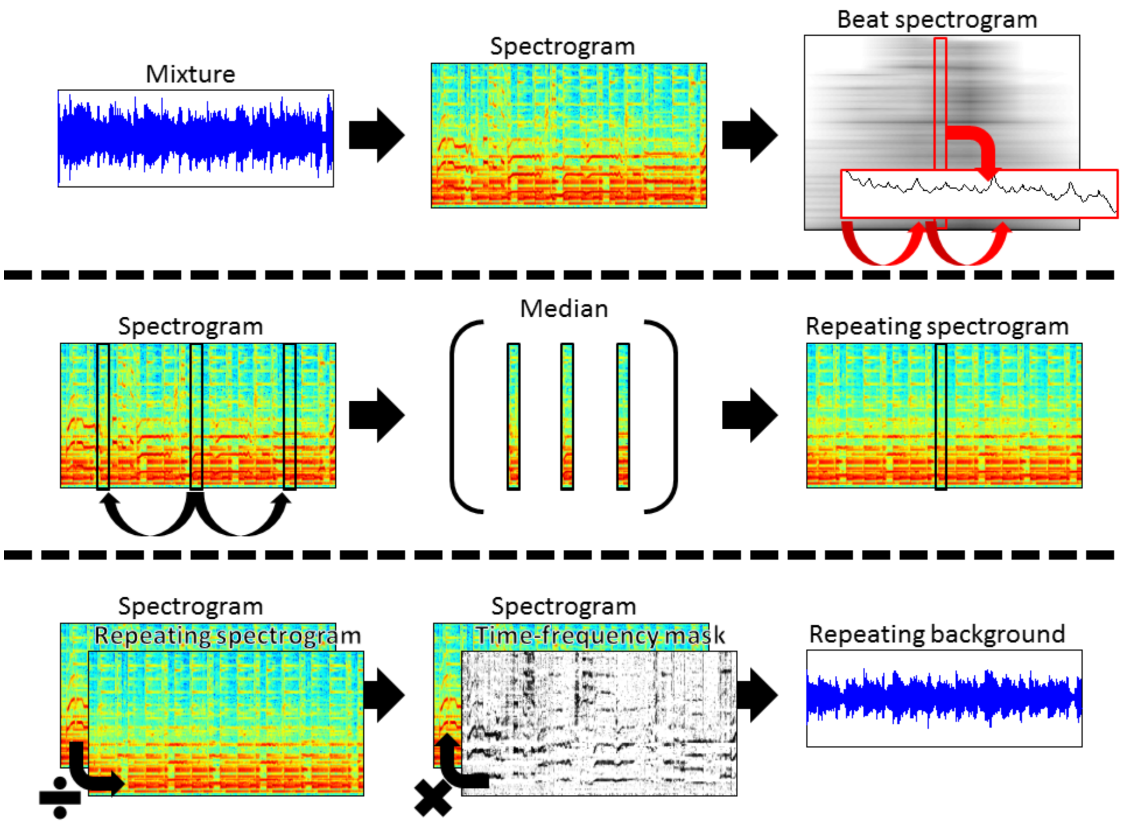

Figure 1.2: Overview of the adaptive REPET: (1) computation of the beat spectrogram and identification of the repeating periods (top row); (2) filtering of the spectrogram and modeling of a repeating spectrogram (middle row); (3) derivation of the time-frequency mask and extraction of the repeating background (bottom row).

the concatenated beat spectra, unveiling the underlying periodically repeating structure of the mixture, as shown in Figure 1.2. If variations of periodicity happen over time in the mixture, the horizontal lines in the beat spectrogram will show variations in their vertical periodicity. The repeating periods can then be identified from the beat spectrogram, for all the time frames in the mixture spectrogram, manually or by using an automatic period finder [23].

Repeating Spectrogram Modeling

In the second stage, the repeating periods are used to model an initial repeating spectrogram by taking, for every time frame in the mixture spectrogram, the median of the time-frequency bins at their period rate, as shown in Figure 1.2

The original REPET assumes that there is a single large period (e.g., the length of a measure) for all the repeating elements in the audio. The adaptive REPET considers each time frame individually (a single frame lasts roughly 20 to 40 milliseconds) and tries to find similar frames separated from the current frame by some period (e.g., 2 seconds). If similar frames are found at this period (e.g., similar frames at -4 seconds, -2 seconds, +2 seconds), the repeating spectrogram model for the current frame is built from those frames. Once this is done, it moves forward to the next time-frame and tries to find a period at which the content of the new frame repeats. This period may or may not be the same as the previous time frame (e.g., a period of 1.9 second instead of 2 seconds). This lets it handle periodically repeating structures that vary slowly over time and also structures that are composed of interleaved repeating patterns of different periods (e.g. minimalist music).

Repeating Structure Extraction

In the third stage, the initial repeating spectrogram is used to derive a refined repeating spectrogram by taking, for every time-frequency bin, the minimum between the initial repeating spectrogram and the mixture spectrogram. The repeating spectrogram is then used to derive a time-frequency mask by dividing, for every time-frequency bin, the repeating spectrogram by the mixture spectrogram, as shown in Figure 1.2.

The repeating background can then be obtained by multiplying, for every time-frequency bins, the time-frequency mask with the STFT of the mixture and transforming the result back to the time-domain. The non-repeating foreground can be obtained by simply subtracting the repeating background from the mixture.

Experiments showed that adaptive REPET can be effectively applied for separating full-track real-world songs (e.g., a whole studio recording) into their accompaniment music and singing voice, unlike the original REPET which would only be meaningful for short excerpts [23].

1.1.3 REPET-SIM

REPET-SIM is a generalization of the REPET approach that was designed to handle non-periodically repeating structures, i.e., when the repeating patterns happen intermittently or without a clear periodicity (e.g., repeated piano stabs in a jazz combo that use the same chord voicing, but whose rhythm varies). This is done by using a similarity matrix to identify the repeating elements [24].

As with the previous two methods, the method can be summarized in three stages (see Figure 1.3): (1) identification of the repeating elements; (2) modeling of a repeating spectrogram; and (3) extraction of the repeating structure.

Repeating Elements Identification

In the first stage, the signal is transformed into a spectrogram. To identify similarities in the mixture, the similarity matrix [25] is derived from the spectrogram by computing the cosine similarity between any two pairs of time frames.

If repeating elements are present in the mixture, the similarity matrix would form regions of high and low similarity at different time indices, unveiling the underlying repeating structure of

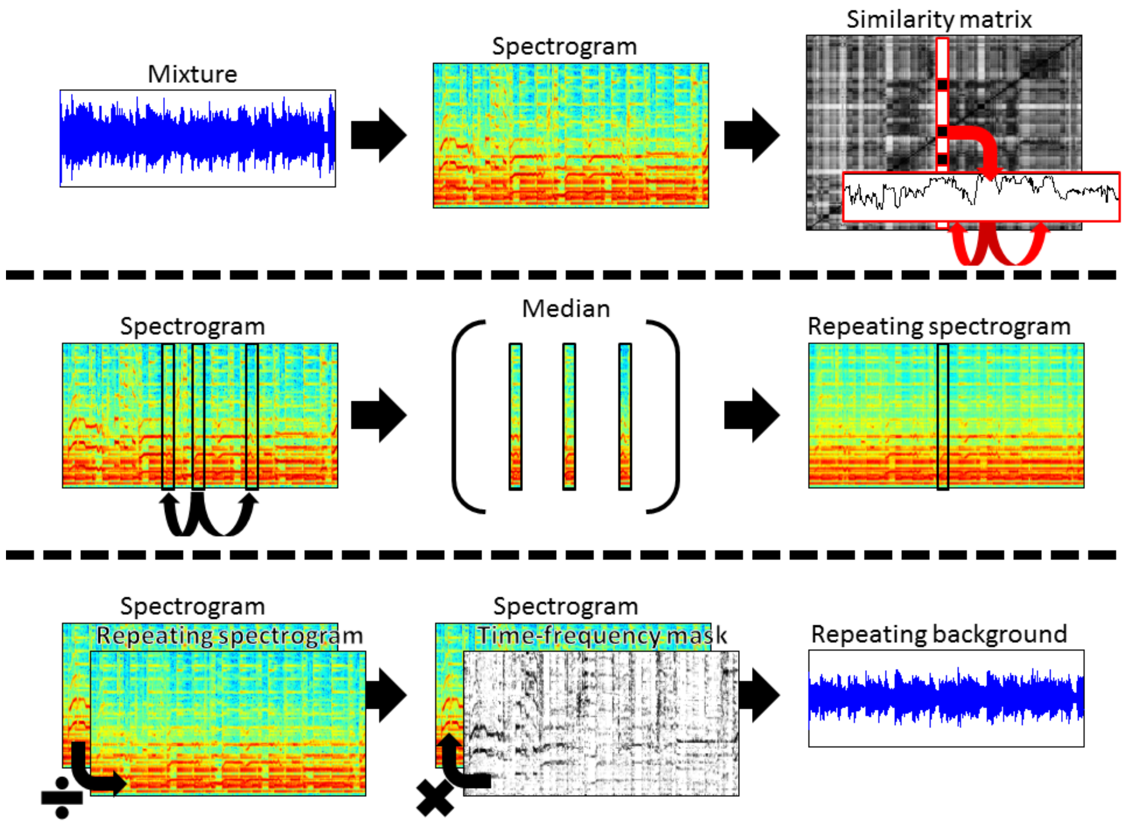

Figure 1.3: Overview of REPET-SIM: (1) computation of the similarity matrix and identification of the repeating elements (top row); (2) filtering of the spectrogram and modeling of a repeating spectrogram (middle row); (3) derivation of the time-frequency mask and extraction of the repeating background (bottom row).

the mixture, as shown in Figure 1.3. The repeating elements can then be identified from the similarity matrix, for all the time frames in the mixture spectrogram, manually or by using an automatic peak finder [24].

Repeating Spectrogram Modeling

In the second stage, the repeating elements are used to model an initial repeating spectrogram by taking, for every time frame in the mixture spectrogram, the median of the time-frequency elements at their repetition rate, as shown in Figure 1.3.

Compared with the other REPET methods (original and adaptive) that look for periodic similarity between events in the audio scene, REPET-SIM looks only for similarity, so that it can handle non-periodically repeating structures, i.e., when the repeating patterns happen intermittently or without a clear periodicity.

Repeating Structure Extraction

In the third stage, the initial repeating spectrogram is used to derive a refined repeating spectrogram by taking, for every time-frequency bins, the minimum between the initial repeating spectrogram and the mixture spectrogram.

The repeating spectrogram is then used to derive a time-frequency mask by dividing, for every time-frequency bin, the repeating spectrogram by the mixture spectrogram, as shown in Figure 1.3.

The repeating background can then be obtained by multiplying, for every time-frequency bin, the time-frequency mask with the STFT of the mixture and transforming the result back to the time-domain. The non-repeating foreground can be obtained by simply subtracting the repeating background from the mixture.

Experiments showed that REPET-SIM can be effectively applied for separating full-track real-world songs into their accompaniment music and singing voice, also compared with the Adaptive REPET [24]. Experiments also showed that REPET-

SIM can be effectively applied for separating two-channel mixtures of one speech source and real-world background noise into their background noise and clean speech, by applying the method online for real-time computing [26].

Note that FitzGerald proposed a method very similar to REPET-SIM for music/voice separation [27].

1.2 Pitch-based Source Separation

The previous sections focused on performing source separation using cues related to the repetitive, rhythmic structure of the music audio. Another fruitful approach is to use the melodic content as embodied by the pitches present in the audio. We now turn to this approach.

Pitched musical instruments (e.g., brass, woodwinds, strings, and keyboard instruments) are harmonic sound sources. Most of the energy in a harmonic sound is located at frequencies that are integer multiples of the fundamental frequency (F0). For example, if a piano plays a single note at A = 440 Hz, most of the energy in the sound will be concentrated at 440 Hz, 880 Hz, 1320 Hz, and so on. While F0 is a physical attribute and pitch is a perceptual attribute of a sound, for a harmonic sound, its pitch can be reliably matched to the F0. Therefore, we do not differentiate F0 from pitch in our discussions here.

Pitch information helps to separate harmonic sources out from a mixture of sounds. In this section, we present our work on pitch-based source separation of audio mixtures composed of harmonic sound sources. Typically, this means separation of instruments from a music recording containing multiple concurrent instruments.

The first step is to estimate the pitches of these harmonic sources from the mixture [49]. This is called *multi-pitch estimation* (MPE). A frame is typically a 20 to 40 millisecond time window. MPE is typically performed in each individual time frame of the audio mixture. The second step is to connect the pitch estimates in different frames to form pitch trajectories, each of which corresponds to a source [52]. This is called *multi-pitch streaming.* The last step is to extract the harmonics from the pitch estimates of each source and reconstruct the source signal. In the following, we describe the three steps separately.

1.2.1 Multi-pitch Estimation

Multi-pitch estimation is the task of estimating pitches and the number of pitches in each frame of a harmonic sound mixture. In music information retrieval, the number of pitches is also called *polyphony.* Many methods have been proposed in the literature. Some do not employ any preprocessing of the signal and work with the full time domain or frequency domain signal [28, 29, 30, 31, 32, 33, 34, 35, 36, 37]; others reduce the signal to a more compact representation such as employing an auditory filterbank as the front end [38, 39, 40, 41] or representing the spectrum only with significant peaks [42, 43, 44]. Our method lies in the second category. More specifically, we represent the power spectrum of the audio mixture with both significant peaks and non-peak regions, and propose a maximum likelihood method to estimate the pitches and the polyphony from the peak/non-peak-region representation [49].

Peak Detection

For harmonic sounds, significant spectral peaks ideally correspond to harmonics of the pitches. Therefore, detection of these peaks would help us infer the pitches. Taking one frame of the audio signal, we first perform Fourier transform [45] to turn the audio into a spectral representation. The middle top panel of Figure 1.1 shows a spectrogram. Here, each column of the image is the spectral representation of a single time step (typically a 20 to 40 millisecond window).

Spectral peaks at each time slice are detected by the peak detector described in [46]. Basically, there are two criteria that determine whether a local maximum in the spectrum should be labeled as a peak. The first criterion is global: the local maximum should not be less than some threshold (e.g., 50 dB) lower than the maximum of the spectrum across all frequencies in that time step. The second criterion is local: the local maximum should be locally higher than a smoothed version of the spectrum by at least some threshold (e.g., 4 dB). Finally, the peak amplitudes and frequencies are refined by quadratic interpolation [47].

We define the *non-peak region* as those frequencies that are further that a quarter tone from any of the detected spectral peaks. Although the non-peak region cannot tell us where the pitches should be, it can tell us where the pitches should not be. Pitches are not likely to be located at frequencies whose harmonics are in the non-peak region.

Likelihood Function

We propose a maximum likelihood method to estimate the set of pitches from the peak/non-peak representation of the mixture spectrum. The likelihood function is defined as the multiplication of the *peak-region likelihood* and the *non-peak-region likelihood*, assuming conditional independence of peaks and the non-peak region given the pitches. The peak region likelihood is defined as the probability of occurrence of the peaks, given an assumed set of pitches. The non-peak region likelihood is defined as the probability of *not* observing peaks in the non-peak region, given an assumed set of F0s. The peak region likelihood and the non-peak region likelihood act as a complementary pair. The former helps find F0s that have harmonics that explain peaks, while the latter helps avoid F0s that would predict harmonics where none are observed (i.e., in the non-peak region).

To define the peak-region likelihood, we characterize each peak by its frequency and amplitude. We further consider the probability that each detected peak is a *normal* peak or a *spu-*

rious peak. By "normal" we mean the peak is generated by some harmonic of the underlying pitches; and by "spurious" we mean the peak is due to other factors such as side-lobes, peak detection errors, etc.

We train the parameters of the likelihood functions using training data with ground-truth pitches and detected peaks and non-peak regions. Two kinds of training data are used. The first kind is a set of isolated notes from the IOWA musical instrument note sample dataset (`http://theremin.music.uiowa.edu/MIS.html`). They are used to train the parameters of the non-peak region likelihood model. The second kind is a set of randomly mixed chords using these isolated notes. They are used to train the parameters of the peak-region likelihood model. Ground-truth pitches are detected using the YIN [48] pitch tracking algorithm on isolated notes before mixing the isolated notes together. Peaks and the non-peak region are detected using the proposed method.

Pitch and Polyphony Estimation

Given the set of detected peaks, frequencies within one semitone around each peak are considered as possible pitches (i.e., possible F0s). The underlying pitches in this frame are thus assumed to compose a subset of these frequencies. This helps to constrain the set of possible hypotheses to reasonable ones, given the data. The task of the maximum likelihood estimation is thus to find the subset of potential F0s that gives the highest likelihood to having generated the observed peaks in the spectrum.

It is, however, intractable to enumerate all these subsets. A typical scene may have 50 to 100 peaks. The set of all subsets of 100 peaks has 2^{100} elements and is not tractable to fully search. We therefore propose a iterative greedy search strategy [49]. We start from an empty subset to represent the estimated pitches. At each iteration, we add the one pitch estimate that most increases the likelihood of the subset. This strategy only enumerates a very small amount of subsets, as the choice of latter pitches depends on the choice of earlier pitches. However, the computational complexity is significantly reduced from exponential to linear with respect to the number of peaks.

An important question for this iterative greedy search process is "When should we stop?" In other words, how many pitches should we estimate in each time frame. Ideally, we hope that the likelihood function can take care of this problem and stop the process when the likelihood does not increase anymore when adding a new pitch. However, similar to many other maximum likelihood estimation problems, there is an overfitting problem. The likelihood typically increases with the number of pitches, although the increase becomes slower. We use a simple thresholding method to address this problem. We do not stop the iterative process until the number of pitches in the subset reaches a predefined maximally possible polyphony (e.g., 9). We then calculate the likelihood increase from 1 pitch to the maximally possible polyphony as the maximally possible likelihood increase. We then estimate the polyphony as the number of pitches that first surpasses 88% of the maximally possible increase. The threshold was tuned on a training set of musical chords with polyphony from 1 to 6. This simple polyphony estimation method is shown to work well on both musical chords and real music pieces.

Pitch Refinement

Pitches and polyphony estimated in individual frames may contain errors that make the pitch contours non-smooth over time. By utilizing contextual information, it is possible to fix these errors and improve the pitch estimation accuracy.

We use a moving average to calculate the refined polyphony estimate with a triangular moving window with size of 19 frames. We also build a pitch histogram with a granularity of a semitone by counting the pitch estimates within the window, and rank the pitches by their weighted counts according to the window. The top several pitches are returned as the refined pitches with equal weights, where the number is equal to the refined polyphony estimate. In experiments we show that this refinement step removes many insertion and deletion pitch estimation errors, and significantly increases the estimation accuracy [49].

1.2.2 Multi-pitch Streaming

The pitches estimated in the previous section can help us separate harmonic sources in each individual frame. However, to separate the source signal over multiple frames, we need to connect these pitches into streams, each of which corresponds to a source. This is the multi-pitch streaming problem. Researchers have proposed to use frequency/amplitude continuity to stream pitches [50, 28, 33, 51], but this approach only works within a note where the pitch does not change abruptly. For streaming pitches into non-continuous pitch contours, we proposed the first method [52]. The basic idea is to use the timbre information. Notes performed by the same instrument have similar timbre than those performed by different instruments. Therefore, we associate each pitch estimate with a timbre feature vector, and perform clustering on the timbre feature vectors. Ideally, each cluster will correspond to one source and their pitches form the pitch stream of that source.

Timbre Features

We need to calculate a timbre feature vector for each pitch estimate in each frame, and this feature should be calculated from the mixture signal directly. In our work we use two kinds of features. The first one is called "harmonic structure" described in [46]. It is defined as a vector of relative logarithmic amplitudes of the harmonics of a pitch estimate. The harmonics are at integer multiples of the pitch. We use the first 50 harmonics to create a 50-dimensional timbre vector. We choose this dimensionality because most instruments have less than 50 prominent harmonics. For each harmonic, we use the peak-finder from [46] to see if there is a significant peak within a musical quarter-tone. If no peak is associated, the magnitude of the harmonic is set to 0 dB, otherwise it is set to the value of the nearest peak. Then, the representation is normalized. This feature has shown to be similar for notes played by the same instrument within a narrow pitch range, while different for different instruments.

Another feature is called the "uniform discrete cepstrum (UDC)" [52]. It is calculated by taking the discrete cosine transform of a sparse log-amplitude magnitude spectrum where the nonzero elements are the harmonics of the pitch. This feature lies in the cepstral feature category and represents the spectral envelope of the harmonics of the target pitch. However, unlike other cepstral features such as the ordinary cepstrum or mel-scale cepstral coefficients (MFCC), UDC can be calculated from the mixture spectrum directly for the target source, without resorting to source separation. UDC has been shown to outperform other cepstral representations and the harmonic structure feature in an instrument recognition task of musical chords [53].

Constrained Clustering

Given the feature vector of each pitch estimate, we perform K-means clustering on the pitches, to minimize the timbre inconsistency within each cluster. However, results show that there are two kinds of common errors. In the middle panel of Figure 1.4, a number of pitches are clustered into the wrong trajectory. For example, the pitches around MIDI number 55 from 14.8 sec to 15.8 sec form a continuous contour and are all played by the bassoon. However, in the clustering, some of them are assigned to saxophone. In another example, from 16.8 sec to 17.6 sec, the K-means clustering puts two simultaneous pitches into the saxophone stream. This is not reasonable, since saxophone is a monophonic instrument.

If we know that different sources do not often perform the same pitch at the same time and

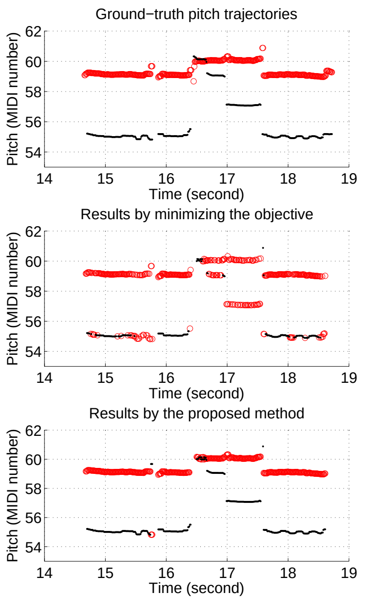

Figure 1.4: Comparison of the ground-truth pitch streams, K-means clustering ($K = 2$) results (i.e., only minimizing the objective function), and the proposed method's results (i.e., considering both objective and constraints). Both the K-means and the proposed method take the ground-truth pitches as inputs, use 50-d harmonic structure as the timbre feature, and randomly initialize their clusterings. Each point in these figures is a pitch. Different instruments are marked with different markers (circles for saxophone and dots for bassoon).

all sources are monophonic, we can impose two kinds of constraints on some pairs of the pitches to improve clustering: A *must-link* constraint is imposed between two pitches that differ less than Δ_t in time and Δ_f in frequency. It specifies that two pitches close in both time and frequency should be assigned to the same cluster. A *cannot-link* constraint is imposed between two pitches in the same frame. It specifies that two simultaneous pitches should be assigned to different clusters. These must-links and cannot-links form the set of all constraints. The bottom panel of Figure 1.4 shows the result obtained from our proposed algorithm, considering both the objective and constraints.

Iterative Algorithm

The formulated constrained clustering problem has two properties that mean existing constrained clustering algorithms [54, 55, 56] cannot be applied: 1) constraints are inconsistent to each other as they are imposed on pitch estimates which contain errors; 2) almost every pitch estimate is involved in some constraint. We therefore propose a new algorithm. It starts from an initial partition that satisfies a subset of all the constraints. Then it iteratively minimizes the objective function while incrementally satisfying more constraints. While the details of the algorithm can be found in [52], here we state its two important properties: 1) it always converges; 2) the timbre inconsistency objective function monotonically strictly decreases while the number of satisfied constraints monotonically increases.

1.2.3 Constructing Harmonic Masks

Given the estimated pitch stream for each harmonic source, we build a soft frequency mask around its harmonics to separate its magnitude spectrum from the mixture spectrum [57]. We then combine the separated magnitude spectrum with the phase spectrum of the mixture signal and perform an inverse Fourier transform to calculate the time-domain signal. Finally, the overlap-add technique [45] is applied to concatenate the current frame to previously separated frames.

To calculate the frequency masks for sources, we first identify their harmonics and overlapping situations from the estimated pitches. Each frequency bin is then classified into three kinds according to the number of harmonics that involve this frequency bin: *nonharmonic bin*, *non-overlapping harmonic bin*, or *overlapping harmonic bin*. For a nonharmonic bin, the masks are designed to evenly distribute the mixture energy to all active sources. For a non-overlapping harmonic bin, the mixture energy is solely distributed to the source whose harmonic involves the bin.

The mask design for overlapping harmonic bins is the most difficult. One can calculate an average harmonic structure template for each source from the harmonic structures of its estimated pitches, and then use the template to design the mask value at different harmonics [46]. This method considers the timbre model of the sources. A simpler method is to distribute the mixture energy to overlapping harmonics, in the inverse proportion to the square of the harmonic indices. This method does not model the timbre of sources, instead, it makes a general assumption that the harmonic amplitude decays at the same rate with respect to the harmonic index, regardless of pitch and instrument that produced the note. This is a very coarse assumption but it provides a simple and relatively effective way to resolving overlapping harmonics.

1.3 Leveraging the Musical Score

Previous sections described source separation algorithms that depend on repeating (rhythmic) content and pitch-based (melodic) content to perform separation. We now consider the case where one can improve a source separation algorithm by using information in addition to the audio recording. For many music pieces, music scores are widely available. In this scenario, score information can be leveraged to help analyze and separate the audio signal. More specifically, if the audio performance is faithful to the score and the audio and the score are aligned (i.e., synchronized), the score can tell us what pitches are likely being played at a time of the audio. This can greatly improve the accuracy of melodic, pitch-based source separation. We can use the score-provided pitch information to separate the harmonic sources in the audio. Such a system is called a score-informed source separation system.

In this chapter, we describe our work on an online score-informed source separation system called *Soundprism* [57]. Soundprism first aligns the audio with the score, then estimates the precise pitches based on the score-provided pitches in each frame, and finally separates audio sources in the frame. All of these operations are performed in an online fashion, i.e., it processes the audio frame-by-frame in a serial fashion, without having the entire audio available. Figure 1.5 illustrates the system overview of Soundprism.

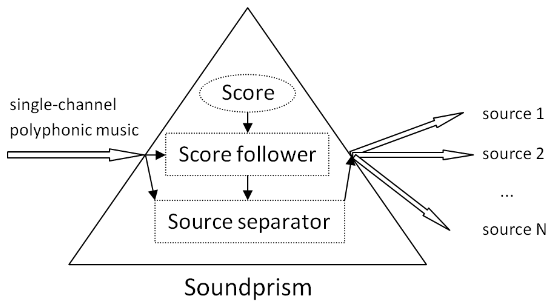

Figure 1.5: System overview of Soundprism.

1.3.1 Audio-score Alignment

The first stage of Soundprism is online audio-score alignment, also called *score following*. The score follower takes a piece of polyphonic music audio as input and pre-stores its electronic score (e.g., MIDI). It then outputs a score position for each time frame in a sequence. We propose a hidden Markov process model to do so, as illustrated in Figure 1.6. The n-th time frame of the audio is associated with a 2-dimensional hidden state vector $\mathbf{s}_n = (x_n, v_n)^T$, where x_n is its score position (in beats) and v_n is its tempo (in Beats Per Minute (BPM)). Each audio frame is also associated with an observation variable $\mathbf{y}_n$, which represents the magnitude spectrum of the time frame. Our aim is to infer the current score position x_n from current and previous observations $\mathbf{y}_1, \cdots, \mathbf{y}_n$. To do so, we need to define a process model to describe how the states transition, an observation model to evaluate hypothesized score positions for the current audio frame, and to find a way to do the inference in an online fashion.

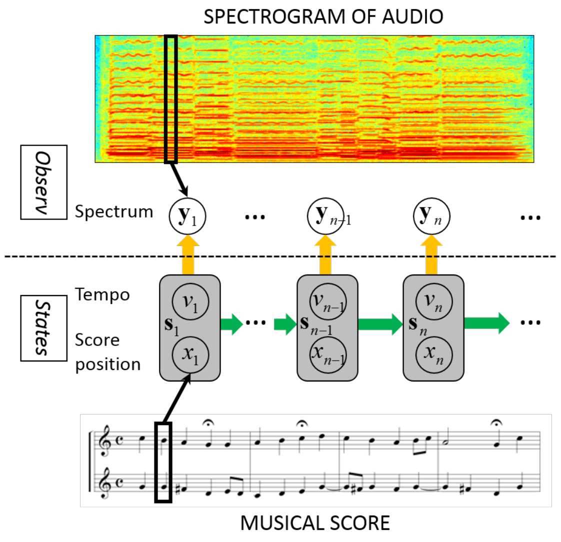

Figure 1.6: Illustration of the state space model for online audio-score alignment.

Process Model

A process model defines the transition probability from the previous state to the current state, i.e., $p(\mathbf{s}_n|\mathbf{s}_{n-1})$. This tells us how the score position and tempo changes from one frame to another and the probability of the change. We use two dynamic equations to define this transition. While the detailed equations are ignored here, the basic idea is that the change of score position is determined by the tempo and the time interval between two adjacent frames. Also the tempo changes randomly around the previous tempo assuming a Gaussian distribution if the score position just passed a note onset or offset, and does not change otherwise.

Note that we do not introduce randomness directly in score position. This is to avoid disruptive changes of score position estimates. In addition, randomness is only introduced when the score position has just passed a note onset or offset. This is because it is rather rare that the performer changes tempo within a note. Second, on the listener's side, it is impossible to detect the tempo change before hearing an onset or offset, even if the performer does make a change within a note. Therefore, changing the tempo state in the middle of a note is not in accordance with music performance, nor does it have evidence to estimate this change.

Observation Model

The observation model evaluates how well a hypothesized state (score position and tempo) can explain the observation, i.e., $p(\mathbf{y}_n|\mathbf{s}_n)$. In this work, we use the multi-pitch likelihood model as described in Section 1.2.1. The basic idea is that if the score pitches at the hypothesized score position fit well to the magnitude spectrum of the current frame $\mathbf{y}_n$, then the hypothesis is good. To calculate the multi-pitch likelihood, we just plug the score pitches into the likelihood function described in Section 1.2.1.

Clearly the observation model itself is not enough to estimate the hidden state (e.g., the score position), as the score may show the same pitches at different positions. In addition, the tempo dimension of the state does not play in the observation model. These problems, however, are addressed when the observation model and process model works together. The process model, considering the tempo dimension and the continuity of state changes, would only favor hypothesized states whose score position is close to the previous score position. This is the key idea of the hidden Markov process model.

Our observation model only considers information from the current frame, and could be improved if considering information from multiple frames. Ewert et al. [58] incorporate inter-frame features to utilize note onset information and improve the alignment accuracy. Joder et al. [59] propose an observation model which uses observations from multiple frames for their conditional random field-based method. In the future we want to explore these directions to improve our score follower.

Inference

Given the process model and the observation model, we want to infer the state of the current frame from current and past observations. From a Bayesian point of view, this means we first estimate the posterior probability $p(\mathbf{s}_n|\mathbf{Y}_{1:n})$, then decide its value using some criterion like maximum a posterior (MAP) or minimum mean square error (MMSE). For hidden Markov processes, this posterior probability at the current frame can be updated from the previous frame. Therefore, we can estimate the posterior probability and the hidden states in an online fashion.

Here we use a *bootstrap filter*, one variant of particle filters [60, 61] to do the online update of the posterior probability. The process starts from an initialization of M particles. Their score positions are all set to the beginning of the score and their tempi are uniformly distributed between half and twice of the score-notated tempo. These particles represent an initialization of the posterior probability. To update the posterior probability when a new audio frame comes in, the particles are first moved using the process model, i.e., the score positions and tempi of the particles are changed. Then the observation likelihood of each particle is calculated using the observation model, which indicates the fitness of the particle to the current audio frame. The likelihood is set as the weight of the particle. These particles are then resampled with replacement according to their weights to generate a new set of M particles. Particles that do not fit to the current frame are less likely to remain in the new particle set, while particles that are a better fit to the current frame may retain multiple copies in the new particle set. A small random perturbation is imposed on these copies to prevent degeneracy of the particles. Now the new set of particles represent the new posterior probability. The average value of these particles is output as the estimate of the hidden state in the current frame.

The set of particles is not able to represent the distribution if there are too few, and is timeconsuming to update if there are too many. In our work we tried to use 100, 1,000 and 10,000 particles. We find that with 100 particles, the score follower is often lost after a number of frames. But with 1000 particles, this rarely happens and the update is still fast enough. Therefore, 1000 particles are used in this chapter.

1.3.2 Pitch Refinement and Source Separation

Given the aligned score position for each audio frame, we know what instrument is playing what pitch in this frame from the score. This important information for separating harmonic sources. However, the pitches provided by the score are integer MIDI pitch numbers. MIDI pitch numbers indicate keys on the piano keyboard. Typically, MIDI 69 indicates the A above Middle C. Assuming A440-based equal temperament allows translation from MIDI pitch to frequency in Hz. The resulting frequencies are rarely equal to the real pitches played in an audio performance. In order to extract the harmonics of each source in the audio mixture, we need to refine them to get accurate estimates of pitches played in the audio.

We refine the pitches using the multi-pitch estimation algorithm as described in Section 1.2.1, but restricting the search space within a semitone of the score-notated pitches. We also assume polyphony is given by the score.

Given the refined pitches in each audio frame, we use the same method described in Section 1.2.3 to construct a harmonic mask for each pitch to separate the magnitude spectrum. Finally, we apply inverse Fourier transform with the phase spectrum of the mixture signal and overlap-add technique to reconstruct the separated source signals.

1.4 Conclusions

In this chapter we have outlined how to perform audio source separation on music using the cues of repeating musical structure and pitch content.

We then showed how one can augment pitch-based separation algorithm by leveraging the information in a musical score, where available. Moving forward, we envision a combination of the score-informed pitch-based separation with separation based on rhythmic structure. Combining these should allow separation of musical instruments from an audio mixture in a large variety of contexts that are currently intractable.

Bibliography

[1] P. Common, C. Jutten: Handbook of Blind Source Separation: Independent Component Analysis and Applications (2010)

[2] T. Virtanen: Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria, IEEE Transactions on Audio, Speech, and Language Processing **15(3)**, 1066 – 1074 (March, 2007)

[3] D. FitzGerald, M. Cranitch, E. Coyle: Nonnegative Tensor Factorisation for Sound Source Separation, Irish Signals and Systems Conference, Dublin, Ireland September 1-2, 2005,

[4] P. Smaragdis, B. Raj, M. V. S. Shashanka: A Probabilistic Latent Variable Model for Acoustic Modeling, NIPS Workshop on Advances in Modeling for Acoustic Processing, Whistler, BC, Canada December 9, 2006,

[5] P.-S. Huang, S. D. Chen, P. Smaragdis: Singing-Voice Separation from Monaural Recordings using Robust Principal Component Analysis, 37th International Conference on Acoustics, Speech and Signal Processing, Kyoto, Japan March 25-30, 2012,

[6] Heinrich Schenker: *Harmony*, Vol. 1 (University of Chicago Press, 1980)

[7] N. Ruwet, M. Everist: Methods of Analysis in Musicology, Music Analysis **6(1/2)**, 3 – 9+11 – 36 (March-July 1987)

[8] A. Ockelford: Repetition in Music: Theoretical and Metatheoretical Perspectives **13 of Royal Musical Association monographs** (2005)

[9] M. A. Bartsch: To Catch a Chorus Using Chroma-based Representations for Audio Thumbnailing, IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, NY, USA October 21-24, 2001,

[10] M. Cooper, J. Foote: Automatic Music Summarization via Similarity Analysis, 3rd International Conference on Music Information Retrieval, Paris, France October 13-17, 2002,

[11] G. Peeters: Deriving Musical Structures from Signal Analysis for Music Audio Summary Generation: "Sequence" and "State" Approach, Computer Music Modeling and Retrieval **2771**, 143 – 166 (2004)

[12] J. Foote: Automatic Audio Segmentation Using A Measure of Audio Novelty, IEEE International Conference on Multimedia and Expo, New York, NY, USA July 30-August, 2000,

[13] J. Foote, S. Uchihashi: The Beat Spectrum: A New Approach to Rhythm Analysis, IEEE International Conference on Multimedia and Expo, Tokyo, Japan August 22-25, 2001,

[14] K. Yoshii, M. Goto, H. G. Okuno: Drum Sound Identification for Polyphonic Music Using Template Adaptation and Matching Methods, ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing, Jeju, Korea October 3, 2004,

[15] R. B. Dannenberg: Listening to "Naima": An Automated Structural Analysis of Music from Recorded Audio, International Computer Music Conference, Gothenburg, Sweden September 17-21, 2002,

[16] R. B. Dannenberg, M. Goto: Music Structure Analysis from Acoustic Signals, Handbook of Signal Processing in Acoustics **1**, 305 – 331 (2009)

[17] J. Paulus, M. Müller, A. Klapuri: Audio-based Music Structure Analysis, 11th International Society on Music Information Retrieval, Utrecht, Netherlands August 9-13, 2010,

[18] J. H. McDermott, D. Wrobleski, A. J. Oxenham: Recovering Sound Sources from Embedded Repetition, Proceedings of the Natural Academy Science of the United States of America **108(3)**, 1188 – 1193 (January 18, 2011)

[19] A. Bregman, C. Jutten: *Auditory Scene Analysis: The perceptual Organization of Sound* (MIT Press, Cambridge, MA, USA 1994)

[20] Z. Rafii, B. Pardo: A Simple Music/Voice Separation System based on the Extraction of the Repeating Musical Structure, 36th International Conference on Acoustics, Speech and Signal Processing, Prague, Czech Republic May 22-27, 2011,

[21] Z. Rafii, B. Pardo: REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation, IEEE Transactions on Audio, Speech, and Language Processing **21(1)**, 71 – 82 (January, 2013)

[22] Z. Rafii, D. L. Sun, F. G. Germain, G. J. Mysore: Combining Modeling of Singing Voice and Background Music for Automatic Separation of Musical Mixtures, 14th International Society for Music Information Retrieval, Curitiba, PR, Brazil November 4-8, 2013,

[23] A. Liutkus, Z. Rafii, R. Badeau, B. Pardo, G. Richard: Adaptive Filtering for Music/Voice Separation Exploiting the Repeating Musical Structure, 37th International

Conference on Acoustics, Speech and Signal Processing, Kyoto, Japan March 25-30, 2012,

[24] Z. Rafii, B. Pardo: Music/Voice Separation using the Similarity Matrix, 13th International Society for Music Information Retrieval, Porto, Portugal October 8-12, 2012,

[25] J. Foote: Visualizing Music and Audio using Self-Similarity, 7th ACM International Conference on Multimedia, Orlando, FL, USA October 30-November 5, 1999,

[26] Z. Rafii, B. Pardo: Online REPET-SIM for Real-time Speech Enhancement, 38th International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada May 26-31, 2013,

[27] D. FitzGerald: Vocal Separation Using Nearest Neighbours and Median Filtering, 23nd IET Irish Signals and Systems Conference, Maynooth, Ireland June 28-29, 2012,

[28] G. E. Poliner, D. P. W. Ellis: A discriminative model for polyphonic piano transcription, EURASIP Journal on Advances in Signal Processing 2007,

[29] M. Davy, S. J. Godsill, J. Idier: Bayesian analysis of polyphonic Western tonal music, Journal of the Acoustical Society of America **119**, 2498 – 2517 (2006)

[30] E. Vincent, M. D. Plumbley: Efficient Bayesian inference for harmonic models via adaptive posterior factorization, Neurocomputing **72**, 79 – 87 (December 2008)

[31] K. Kashino, H. Murase: A sound source identification system for ensemble music based on template adaptation and music stream extraction, Speech Communication, 337 – 349 (1999)

[32] M. Goto: A real-time music-scene-description system: predominant-F0 estimation for detecting melody and bass

lines in real-world audio signals, Speech Communication **43**(4), 311 – 329 (2004)

[33] H. Kameoka, T. Nishimoto, S. Sagayama: A multipitch analyzer based on harmonic temporal structured clustering, IEEE Trans. on Audio Speech and Language Processing **15**(3), 982 – 994 (2007)

[34] S. Saito, H. Kameoka, K. Takahashi, T. Nishimoto, S. Sagayama: Specmurt analysis of polyphonic music signals, IEEE Trans. Speech Audio Processing **16**(3), 639 – 650 (2008)

[35] J.-L. Durrieu, G. Richard, B. David: Singer melody extraction in polyphonic signals using source separation methods, Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2008, 169 – 172

[36] V. Emiya, R. Badeau, B. David: Multipitch estimation of quasi-harmonic sounds in colored noise, Proc. International Conference on Digital Audio Effects (DAFx) 2007,

[37] G. Reis, N. Fonseca, F. Ferndandez: Genetic algorithm approach to polyphonic music transcription, Proc. IEEE International Symposium on Intelligent Signal Processing 2007,

[38] T. Tolonen, M. Karjalainen: A computationally efficient multipitch analysis model, IEEE Trans. on Speech and Audio Processing **8**(6), 708 – 716 (2000)

[39] A. de Cheveigné, H. Kawahara: Multiple period estimation and pitch perception model, Speech Commun. **27**, 175 – 185 (1999)

[40] A. Klapuri: Multiple fundamental frequency estimation based on harmonicity and spectral smoothness, IEEE Trans. Speech Audio Processing **11**(6), 804 – 815 (2003)

[41] A. Klapuri: Multiple fundamental frequency estimation by summing harmonic amplitudes, Proc. ISMIR 2006, 216 – 221

[42] R. J. Leistikow, H. D. Thornburg, J. S. Smith, J. Berger: Bayesian identification of closely-spaced chords from single-frame STFT peaks, Proc. International Conference on Digital Audio Effects (DAFx'04), Naples, Italy 2004, 228 – 233

[43] A. Pertusa, J. M. Inesta: Multiple fundamental frequency estimation using Gaussian smoothness, Proc. IEEE International Conference on Acoustics, Speech Signal Processsing (ICASSP) 2008, 105 – 108

[44] C. Yeh, A. Röbel, X. Rodet: Multiple fundamental frequency estimation of polyphonic music signals, Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2005, 225 – 228

[45] J.O. Smith: *Spectral Audio Signal Processing* (`http://ccrma.stanford.edu/~jos/sasp/`, March 24, 2014)

[46] Z. Duan, Y. Zhang, C. Zhang, Z. Shi: Unsupervised single-channel music source separation by average harmonic structure modeling, IEEE Trans. Audio Speech Language Processing **16**(4), 766 – 778 (2008)

[47] J. O. Smith, X. Serra: PARSHL: an analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation, Proc. Internetional Computer Music Conference (ICMC) 1987,

[48] A. de Cheveigné, H. Kawahara: YIN, a fundamental frequency estimator for speech and music, Journal of the Acoustical Society of America **111**, 1917 – 1930 (2002)

[49] Zhiyao Duan, Bryan Pardo, Changshui Zhang: Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions, IEEE Trans. Audio

Speech Language Processing **18**(8), 2121 – 2133 (2010)

[50] M. Ryynanen, A. Klapuri: Polyphonic music transcription using note event modeling, Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) 2005, 319 – 322

[51] W.-C. Chang, A. W. Y. Su, C. Yeh, A. Robel, X. Rodet: Multiple-F0 tracking based on a high-order HMM model, Proc. International Conference on Digital Audio Effects (DAFx) 2008,

[52] Z. Duan, J. Han, B. Pardo: Multi-pitch streaming of harmonic sound mixtures, IEEE Trans. Audio Speech Language Processing **22**(1), 1 – 13 (2014)

[53] Z. Duan, B. Pardo, L. Daudet: A novel cepstral representation for timbre modeling of sound sources in polyphonic mixtures, Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2014,

[54] K. Wagstaff, C. Cardie: Clustering with instance-level constraints, Proc. International Conference on Machine Learning (ICML) 2000, 1103 – 1110

[55] K. Wagstaff, C. Cardie, S. Rogers, S. Schroedl: Constrained k-means clustering with background knowledge, Proc. International Conference on Machine Learning (ICML) 2001, 577 – 584

[56] I. Davidson, S. S. Ravi, M. Ester: Efficient incremental constrained clustering, Proc. ACM Conference on Knowledge Discovery and Data Mining (KDD) 2007, 240 – 249

[57] Z. Duan, B. Pardo: Soundprism: an online system for score-informed source separation of music audio, IEEE Journal of Selected Topics in Signal Processing **5**(6), 1205 – 1215 (2011)

[58] S. Ewert, M. Müller, P. Grosche: High resolution audio synchronization using chroma onset features, Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 2009, 1869 – 1872

[59] C. Joder, S. Essid, G. Richard: A conditional random field framework for robust and scalable audio-to-score matching, IEEE Trans. Audio Speech Language Processing **19**(8), 2385 – 2397 (2011)

[60] *Sequential Monte Carlo Methods in Practice*, ed. by A. Doucet, N. de Freitas, N. J. Gordon (Springer-Verlag, 2001)

[61] M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp: A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Processing **50**(2), 174 – 188 (2002)