SEPARATING PIANO RECORDINGS INTO NOTE EVENTS USING A PARAMETRIC IMITATION APPROACH

Wen Xue
Samsung Electronics

xue.wen@samsung.com

ABSTRACT

In this paper we present a working system for separating a piano recording into events representing individual piano notes. Each note is parameterized with a transient-plus-harmonics model that, should all the parameters be reliably estimated, would produce near perfect reconstruction for each note as well as for the whole recording. However, interference between overlapping notes makes it hard to estimate parameters from their combination. In this work we propose to assess the estimability of sinusoidal parameters via their apparent degree of interference, estimate the estimable ones using algorithms suitable for different interference situations, and infer the hard-to-estimate parameters from the estimated ones. The outcome is a sequence of separate, parameterized piano notes that perceptually highly resemble, if are not identical to, the notes in the original recording. This allows for later analysis and processing stages using algorithms designed for separate notes.

1. INTRODUCTION

Musical note separation is the task of breaking audio-based musical content into separate pieces of audio, each corresponding to a musical note in the original content, as if recorded separately. By allowing access to individual notes, note separation has immense potential in the production, maintenance and consumption of recorded music. Unfortunately, high-quality fully-automatic audio-based note separation remains hard. More feasible alternatives have been proposed to address the task. For example, automatic score-informed separation, e.g. [1][2], uses the musical score to guide the separation process. Supervised separation, e.g. [3][4], engages human power to provide more detailed and reliable information that the separation module may benefit from.

In this paper we propose another supervised note separator configured for real-world piano recording. Among musical instruments the piano is known to produce note sounds that are more predictable and less volatile therefore easier to describe and measure. Despite this we still have two points to consider before we can separate the complete set of notes from a recording. First, note separation implicitly includes music transcription, which is still an open problem itself; second, piano music is largely polyphonic, and it is common to have severe interference between concurrent notes, which makes clean separation difficult.

Our answer (or concession) to the first point is an interactive, supervised process that relies on a human participant to make decisions and correct errors. For the second point we propose an automated method that makes measurements where interference is low, and guesses where interference is high. Since some parts of the notes are inferred rather than measured, the method does not qualify for signal separation in the strict sense. We call it a parametric imitation approach, as it imitates an original note with incomplete measurements. By not measuring the hard-to-measure parts of the signal, this scheme minimizes the risk of unstable estimates from high-interference zones. This allows resynthesis of notes that 1) resemble the original ones in loudness, pitch, timbre and dynamics, and 2) sound convincing by themselves.

Our note separator works in four stages:

1. transcription, for finding out what notes are in the music and decide their timing and component frequencies;
2. interference classification, for finding high, mid and low interfering zones in the time-frequency (T-F) plane;
3. stationary component estimation, for estimating note parameters in low and mid interference zones and guessing them in high interference zones;
4. transient extraction and note reconstruction.

Stage 1 is semi-automatic with limited human participation; the rest are fully automatic.

The rest of this paper is arranged as follows. Section 2 describes the signal model we use to represent piano notes. Section 3 describes the user interface in the transcription stage. Sections 4, 5, 6 and 7 present the algorithms in the four stages above, respectively. Section 8 includes experimental result that highlights our performance for interfering notes. Future improvements are discussed in section 9.

2. MODEL AND ASSUMPTIONS

For underlying signal model we use a transient-plus-harmonics model similar to [5]. The transient part models the attack of a note; the harmonic part models the pitched stationary resonance. We assume that piano notes have constant pitch after the initial attack, so the harmonics part $x(t)$ of a note can be written as

$$x(t) = \sum_{m=1}^{M} a^m(t)\cos(2\pi f^m t + \varphi^m) \tag{1}$$

where M is the number of component sinusoids (partials). $a^1(t)$, …, $a^M(t)$ are the amplitudes of the sinusoids and f^1, …, f^M are their frequencies. Functions $a^1(t)$, …, $a^M(t)$ are constrained to vary slowly with t. In this paper we use superscripts for partial indices. To distinguish them from exponents, we write the latter with brackets on the base, like $(m)^2$ or $|X|^2$.

Many pitched instruments have all partial frequencies determined by the fundamental frequency via a harmonicity (or equivalently, "inharmonicity") model:

$$f^m = f^m(m, \mathrm{F0}; \bullet) \tag{2}$$

where m is the partial index, F0 is the fundamental frequency and $\bullet$ represents optional parameters. For the piano we choose the stiff string model in [6], plus to a small additional shift:

$$f^m(m, \mathrm{F0}; B) = m \cdot \mathrm{F0}\sqrt{1 + B\left((m)^2 - 1\right)} + \varepsilon^m \quad (3)$$

where B is a small positive number representing string stiffness, and ε^m covers inharmonicity due to other factors. In section 4 we estimate F0 and B by minimizing these ε^m's.

While the vibrating modes of an ideal wave-radiating string are characterized by exponentially decaying amplitudes, such is not suitable for modelling partial amplitudes in (1). This is because modern pianos use 2-string and 3-string notes, producing amplitude modulation typical of beating sinusoids. This prevents us from using parametric method like ESPRIT [7] for estimation. Fortunately, in the low to mid frequency range where most energy lies, this amplitude modulation is usually slow enough to be effectively captured with a uniformly-sampled sinusoidal representation like [5]. Given the complexity of real-world recordings, we make no special assumption for measuring amplitude parameters except that they vary slowly with time, and that they decay in the long term.

In this paper we evaluate all parameters from the short-time Fourier transform (STFT) of the recording, computed using a Hann window 2048 points long applied to waveform data sampled at 44.1kHz. Adjacent windows overlap by 50%, which makes the hop size 23.2 milliseconds. The parameter set for each note includes the position of starting frame (1), length in frames L (1), number of partials M (1), frequency of each partial (M), initial phase angle of each partial (M), amplitude of each partial at each frame (LM) and transient spectrum (2048).

3. USER INTERACTION IN SUPERVISED TRANSCRIPTION AND FREQUENCY ESTIMATION

The goal of the transcription stage (stage 1) is to find out what notes are in the recording, where they begin and end, and what their partial frequencies are. While note separation necessarily includes music transcription as subtask, this paper is not about automatic transcription. Instead, we follow the supervised path and involve a human user, the *supervisor*, who provides information hard to retrieve reliably using present automatic techniques. More specifically, in this paper the user's job is to help the computer locate harmonic partials of each note in the T-F plane using the spectrogram. To reduce his workload it is recommended that an automatic transcription system like [8]-[11] be used as front end to provide initial guesses of existing notes and their whereabouts. The proposed workflow does not tell if the initial guess comes from a human user or an automatic transcriber. Subsequent steps will require user participation to clean up any mistakes previously made.

The workflow of our note separator is shown in Figure 1. The shaded blocks are automated modules and the clear ones need user attention. The block labelled "supervisor input" may also include an automatic transcription front end, if there is one.

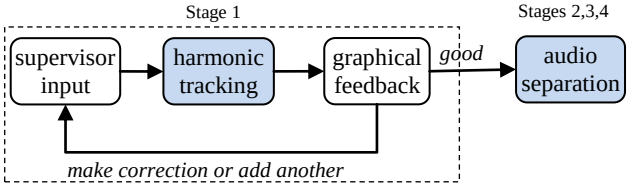


Figure 1 Note separation workflow

The core module of stage 1 is an automatic harmonics tracker (section 4) that locates all harmonics of a note in the T-F plane. For each note, the tracker takes one or more *seed* points as input. A seed point comprises a time-frequency-partial_index triple (t, f, m), meaning that the m^{th} partial of the note has frequency near f at time t. The initial seed point can be provided either by an automatic front end or by the human user. To supply the seed without automatic transcription, a graphical user interface with an image of the spectrogram is presented to the user. The user identifies a note picking a partial index m then pointer-clicking on an unambiguous point of the m^{th} partial in the image. The partial index and the coordinates of the pointer click make up the seed point, with which the automatic tracker is launched. The tracker tolerates no less than two bins of input frequency inaccuracy, so that the average user can supply seed points with comfort.

The tracking result is fed back to the user on the same user interface, with note duration and partial frequencies plotted as frequency trails on the spectrogram, like in Figure 1. The hollow star in the figure marks the initial guess where the user has seeded the harmonics tracker. The user can judge if the tracking has been successful by comparing the trails against the spectrogram.

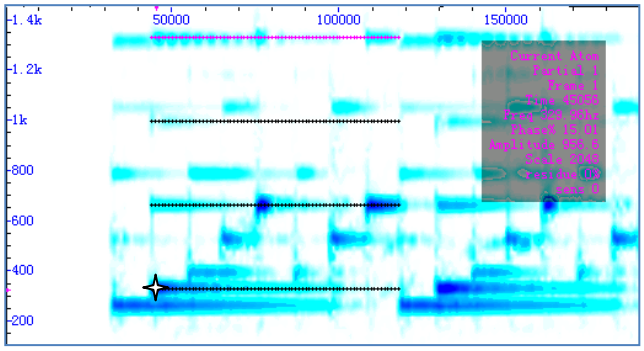


Figure 2 Selecting a note on the spectrogram

Three types of mistakes may occur during automatic tracking: wrong partial frequency, wrong start position and wrong end position. Start and end position errors can be corrected by the user fixing them directly on the spectrogram. Frequency mistakes typically involve frequency estimates being associated with harmonics from other notes. The user corrects them by adding one more seed point, which strengthens the harmonicity constraint and forces estimates off the wrong sinusoids. Our user interface allows the user to drag a frequency trail to another position using a pointer device, upon which the harmonics tracker is relaunched with an additional seed point at the new position. Each error correction requires one pointer clicking or dragging operation. In our experiment a prominent piano note takes no more than 2 operations to mark (including the initial guess), while a heavily masked note usually takes 3 or 4. This track-and-correct procedure is repeated for all notes we intend to separate, plus all their concurrent notes. The rest is left to the note separation module that does not need supervision.

4. HARMONIC FREQUENCY TRACKING FOR CONSTANT PITCHES

This section discusses the harmonic tracker in the supervised transcription stage (stage 1). Given one or more seed points that identify a note, its task is to automatically find out all partial frequencies of the note, along with its start and end positions.

In standard sinusoidal modelling [12]-[14], the frequency of a sinusoid is measured independently from spectral peaks at individual frames. Such measurement is easily corrupted by noise or interference from other sources. Luckily the constant-frequency and harmonicity constraints allow us to ignore data from areas in the T-F plane where such corruption is high, and use only the less corrupted parts for estimating frequencies.

Our constant-pitch harmonics tracker uses plain spectrogram for audio input. From the spectrogram a collection of spectral *atoms* are obtained using standard peak picking. Given the presence of noise and interference neither constant-frequency tracks nor harmonic atom groups are guaranteed to emerge from these raw atoms. However, it is plausible to assume that at least *some* of the atoms are relatively less corrupted. These atoms should assume spectral shape typical to constant-frequency sinusoids; their frequency estimates should be accurate and satisfy constant-frequency and harmonicity constraints. Our harmonics tracker uses peak shape and the frequency constraints to look for the "good" atoms and use them for frequency estimation. We call them the *core* atoms.

4.1. Peak shape

All spectral atoms identified by peak picking are located at spectral peaks. Given an atom with frequency estimate *f*, in bins, we evaluate the peak shape by its cross-correlation with that of a pure sinusoid at *f*, i.e.

$$\lambda_{\mathrm{p}} = \frac{\sum_k X_k W^*(k-f)}{\left(\sum_k |X_k|^2\right)^{1/2}\left(\sum_k |W(k-f)|^2\right)^{1/2}} \tag{4}$$

in which X_k is the STFT of the signal x at bin k, and $W(f)$ is the discrete-time Fourier transform of the window function used for spectrogram computation. The summations are done over a few bins near f. The value of $|\lambda_p|$ lies on [0, 1] (Cauchy-Schwarz). The higher is $|\lambda_{\mathrm{p}}|$, the closer the atom spectrum resembles that of a sinusoid. The harmonic tracker uses $|\lambda_{\mathrm{p}}|$ to screen spectral atoms as candidates for the core set: only those with $|\lambda_{\mathrm{p}}|$ value higher than a threshold (e.g. 0.9) may become a core atom.

4.2. Constant-frequency constraint

Given a set of L atoms detected from L different frames with frequency estimates $f_0, \ldots, f_{L-1}$, we measure how constant these estimates are by their average deviation from a presumed frequency $\hat{f}$:

$$\lambda_{\mathrm{f}} = \frac{1}{L}\sum_{l=0}^{L-1}\left(D(f_l, \hat{f})\right)^2 . \tag{5}$$

where D is a distance function. Eq. (5) cannot be evaluated without knowing $\hat{f}$. We select both $D(\cdot)$ and $\hat{f}$ to fit into a harmonicity model so that the constant-frequency and harmonicity constraints are combined in one, as described below.

4.3. Harmonicity constraint

For the piano we choose the model given in [6] for stiff strings:

$$\hat{f}^{\,m}(m, \mathrm{F0}, B) = m \cdot \mathrm{F0}\sqrt{1 + B\left((m)^2 - 1\right)} . \tag{6}$$

where B is a small positive number. Given a set of I atoms with frequency estimates $f^{m_0}, \ldots, f^{m_{I-1}}$, m_i being the assumed partial index of the i^{th} atom, we measure their harmonic consistency by their deviation from the harmonicity model:

$$\lambda_{\mathrm{f}} = \frac{1}{I}\sum_{i=0}^{I-1}\left(D(f^{m_i}, \hat{f}^{\,m_i}(\mathrm{F0}, B))\right)^2 \tag{7}$$

The smaller is λ_{f}, the more likely are the atoms to belong to the same note. Evaluating (7) requires knowing the values F0 and B, which we choose by minimizing λ_{f}, as follows.

Eq. (6) is nonlinear regarding F0 and B. We linearized it by taking $F=\mathrm{F0}^2$ and $G=FB$ (also see [14]) as

$$(\hat{f}^{\,m})^2 = (m)^2 F + (m)^2\left((m)^2 - 1\right)G \tag{8}$$

Eq. (8) is linear regarding F and G. We choose the distance function $D(\cdot)$ as:

$$D\left(f^{m_i}, \hat{f}^{\,m_i}\right) = \frac{1}{m_i}\left((f^{m_i})^2 - (\hat{f}^{\,m_i})^2\right) \tag{9}$$

where the multiplier $1/m$ removes extra emphasis put onto high frequencies by the squaring. Substitute (8) and (9) in (7) we get

$$\lambda_{\mathrm{f}} = \frac{1}{I}\sum_{i=0}^{I-1}\frac{1}{(m_i)^2}\left((f^{m_i})^2 - (m_i)^2 F - (m_i)^2((m_i)^2 - 1)G\right)^2 . \tag{10}$$

We find F and G that minimize λ_{f} with

$$F = \frac{\sum (f^{m_i})^2 \sum m_i^2(m_i^2-1)^2 - \sum (f^{m_i})^2 (m_i^2-1)\sum m_i^2(m_i^2-1)}{\sum m_i^2 \sum m_i^2(m_i^2-1)^2 - \sum m_i^2(m_i^2-1)\sum m_i^2(m_i^2-1)},$$

$$G = \frac{\sum m_i^2 \sum (f^{m_i})^2 (m_i^2-1) - \sum (f^{m_i})^2 \sum m_i^2(m_i^2-1)}{\sum m_i^2 \sum m_i^2(m_i^2-1)^2 - \sum m_i^2(m_i^2-1)\sum m_i^2(m_i^2-1)} . \tag{11}$$

Using these values we are able to evaluate λ_{f} by (10). Notice there is no constraint on time or partial index associated with each atom involved, except that they cannot all have the same partial index. If (10) is applied to multiple frames, λ_{f} measures frequency consistency both across time and across partials.

The harmonics tracker does not use λ_{f} directly, but uses F and G instead for screening spectral atoms as candidates for the core set, as detailed below.

4.4. Frequency range of eligible atoms given other atoms

The harmonics tracker needs a set of core atoms from multiple partials and frames to estimate the frequencies. We construct this core atom set by incrementally including new atoms as the tracking progresses. The frequency of a newly incorporated core atom should be consistent with existing core atoms. One way to evaluate this consistency is to estimate F and G from the current core set using (11), then predict the frequency $\hat{f}^{\,m}$ for any partial index m using (6). Only atoms within a δ-vicinity of $\hat{f}^{\,m}$, i.e. ($\hat{f}^{\,m}$ - δ , $\hat{f}^{\,m}$ +δ), as considered eligible candidates for core atoms of the m^{th} partial. Since core atoms are assumed to have accurate frequency estimates, we choose a relatively small vicinity, e.g. δ=0.1 bins.

To be able to estimate F and G above with (11), we must already have at least two core atoms with different partial indices. In the case only one, say f^m, is available, we can determine the eligible range for another partial index by fixing an upper bound for B, say B_{M}. The eligible range for F0 then becomes:

$$\frac{f^m - \delta}{m\sqrt{1 + B_{\mathrm{M}}\left((m)^2 - 1\right)}} < \mathrm{F0} < \frac{f^m + \delta}{m} . \tag{12}$$

This gives an eligible range for the n^{th} partial as

$$\frac{n}{m}\frac{f^m-\delta}{\sqrt{1+B_M\left((m)^2-1\right)}}-\delta<f^n<\frac{n}{m}(f^m+\delta)\sqrt{1+B_M\left((n)^2-1\right)}+\delta\,. \quad (13)$$

4.5. The tracking algorithm

The constant-pitch harmonics tracker proceeds frame-by-frame, from the seed point forwards and backwards until an endpoint is met. We require that the starting point be an actual atom detected with high λ_p, e.g. above 0.9. The tracking algorithm maintains a core atom set **C**, which is initiated as empty. The eligible frequency range for the first atom (when **C** is empty) is defined as a few bins around the seed point.

In each tracking step the tracker moves one frame forward or backward, finds new core atoms from the new frame, and removes existing core atoms that appear no longer good enough. Given the current core atom set **C**, a single-frame tracking step proceeds as follows.

Within the current frame, do
1 ºfind the eligible ranges for all partials consistent with **C**;
2 ºfind all atoms with high λ_p within these eligible ranges, let this set of new atoms be $\mathbf{C}_{\text{new}}$;
3 ºif there are multiple atoms found for any partial index, do
4 %~5 º
4 º use $\mathbf{C}\cup\mathbf{C}_{\text{new}}$ to predict the frequency for that partial index;
5 º keep the atom closest to the predicted frequency in $\mathbf{C}_{\text{new}}$ and remove competing atoms;

With the current set of core atoms, do
6 ºuse $\mathbf{C}\cup\mathbf{C}_{\text{new}}$ to predict the frequencies for all atoms in the core set, remove those that deviate more than δ from the prediction.

From the seeding frame l, forward tracking repeats this step for frames l, l+1, l+2,…, until the number of consistent atoms found falls below a threshold level for three consecutive frames. We currently set the threshold at a fifth of the total number of partials for the relevant pitch, which is computed with (8). Similarly, backward tracking repeats the step for frames l-1, l-2, …, until an end point is met. The tracking algorithm returns the event duration and all partial frequencies estimated from the final core atom set by partial-wise average, weighted by the atom energy. If not enough core atoms are available to compute the average for some partial, its frequency is computed with (11).

While the previous tracking algorithms [12]-[14] also estimate amplitudes and phase angles, doing so without considering interference between concurrent notes leads to faulty results. In this paper we address the interference using what we call the collision regions. We explain them in the next section.

5. COLLISION REGIONS

This section discusses the interference classification stage (stage 2), whose goal is to provide detailed description on the interference between sinusoidal partials, so that the subsequent stage can use this information for estimating amplitudes.

Spectral interference occurs if partials of concurrent notes have very close frequencies. To properly address interference we want to know exactly what partials have what level of interference during which time interval. Each such group of partials has its own characteristics concerning interference and are best treated with an estimator tuned to that special occasion. As interference is caused by concurrency in time and proximity in frequency, such interfering partial groups occupy localized regions in the time-frequency plane.

5.1. Colliding sinusoids

We say two frequencies *collide* if they are closer than a reference threshold δ_f from each other, so that the presence of one may compromise the estimation of the other. The value of δ_f is related to the frequency resolution of the estimator. As we measure sinusoids from STFT, a convenient choice is a fixed value of δ_f in DFT bins. We say two sinusoids collide if their frequencies collide.

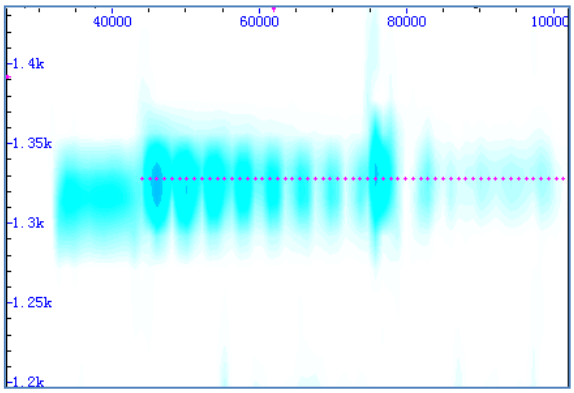


Figure 3 Detail of Figure 2: colliding partials

Figure 3 shows the spectral detail of the 1st half of the signal in Figure 2 near 1.3kHz. At the beginning we have the first note (C4=262Hz) starting around time 32800 (at 44.1kHz sample rate). Its 5th partial has frequency measured at 1316.6Hz. Then at time 44000 the second note (E4) enters. Its 4th partial frequency is 1328.1Hz, which is very close to the 5th partial of the first note. The 11.5Hz difference in their frequencies is too small to be told by the spectrogram, computed with window size 2048 (46.4ms). Consequently we observe a distinctive interference pattern in the central part of Figure 3. Such beating pattern is a typical sign of colliding partials. Since components from different notes cancel one another from time to time, nonnegativity-based methods (e.g. [1]-[3]) cannot handle the case without redesign. It is for the estimation of sinusoids from such spectrograms that we propose the idea of collision regions.

Given a set of constant-frequency sinusoids with known durations and frequencies, it is trivial to determine whether and when any two of them collide. Additional complication arises as more than two sinusoids lie close to each other, all starting and finishing at different times. To describe sinusoid collisions in an organized way suitable for algorithmic handling, we cut the T-F plane into rectangular tiles called collision regions, each containing a number of colliding sinusoids from start to end.

5.2. Definition

We define a *collision region* (CR) of a set **S** of sinusoids as a rectangular area in the time-frequency plane, described by a pair of coordinates (t_1, f_1) and (t_2, f_2), so that:

a) $\forall s \in \mathbf{S}$ have durations containing $[t_1, t_2]$ and frequencies within (f_1, f_2);

b) $f \in (f_1, f_2)$ if and only if there is $s \in \mathbf{S}$ so that f and s collide.

It follows that for every $s \in \mathbf{S}$ there is at least one other $s' \in \mathbf{S}$ that collides with s over $[t_1, t_2]$. We denote a collision region as CR:($\mathbf{S}$; t_1, f_1, t_2, f_2), or simplified as CR($\mathbf{S}$).

A sinusoid $s' \notin \mathbf{S}$ is said to collide with CR:($\mathbf{S}$; t_1, f_1, t_2, f_2) if it collides with any $s \in \mathbf{S}$ at any $t \in [t_1, t_2]$. A collision region of $\mathbf{S}$ is said to be *closed* if no sinusoid outside $\mathbf{S}$ collides with it. Sinusoids in a closed CR($\mathbf{S}$; t_1, f_1, t_2, f_2) are considered free from interference from sinusoids outside $\mathbf{S}$ during $[t_1, t_2]$. If the CR contains multiple sinusoids, their mutual interference should be considered for estimating their parameters during $[t_1, t_2]$.

For example, an isolated sinusoid s of duration $[t_1, t_2]$ and frequency f has its own trivial CR:({s}; t_1, f-δ_f, t_2, f+δ_f) which is also closed. Two colliding sinusoids s_1 and s_2 of duration [0,2] and [1,3] have 3 closed CRs, on intervals [0,1], [1,2] and [2,3], respectively.

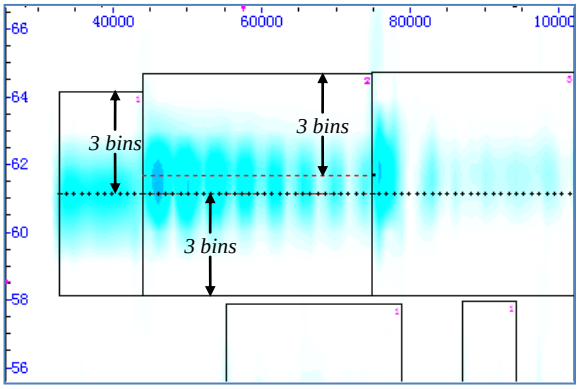


Figure 4 Collision regions of Figure 3

Figure 4 shows the collision regions marked out for the T-F range of Figure 3 with δ_f=3 bins. In Figure 4 we have marked the frequency axis in bins. Between time 32800 and 44000 the 5th partial of the first note is free from interference, therefore it occupies its own CR with total bandwidth of 6 bins, 3 (i.e. δ_f) on each side. Between 32800 and 74800 we have two partials 0.54bin apart. Since 0.54< 3, these two partials are put into the same CR, whose bandwidth becomes 6.54 bins. At 74800 the fifth note starts whose 2nd partial almost overlaps the 4th partial of the second note. This establishes a new CR that contains 3 partials.

5.3. Finding collision regions

Given a set of sinusoids $\mathbf{S}$, we look for a complete set of closed CRs to cover $\mathbf{S}$ using an iterative process. Starting from an arbitrary sinusoid $s_0 \in \mathbf{S}$ with its trivial CR, we add other sinusoids s_1, s_2, …, one at a time into the picture. Each time a new sinusoid s_i is added, we update the set of CRs so that 1) every CR is still closed and 2) the whole CR set covers the new sinusoid as well as the previous ones.

5.3.1. *Collision table*

The update process relies on a data structure we call a *collision table*. Given a closed CR set $\mathbf{C}$ covering sinusoid set $\mathbf{S}$ and a new sinusoid $s \notin \mathbf{S}$, the collision table tells which CRs in $\mathbf{C}$ collide with s at what time. To be more specific, the table $\mathbf{T}(s,\mathbf{C})$ contains a sequence of non-overlapping intervals that together cover the duration of s, so that on each of these intervals s collide with a different combination of 0, 1 or 2 collision regions in $\mathbf{C}$.[1] The collision table is generated by finding all CRs in $\mathbf{C}$ which collide with s, segmenting the duration of s at their boundaries, and enumerating the 0, 1 or 2 CRs that collide with s over each segment.

5.3.2. *Updating the complete CR set*

Let $\mathbf{C}$ be a closed CR set covering $\mathbf{S}_{i-1}$, and $\mathbf{T}(s_i,\mathbf{C})$ be the collision table computed for the next sinusoid s_i. The following routine updates $\mathbf{C}$ to a closed CR set covering $\mathbf{S}_i = \mathbf{S}_{i-1} \cup \{s_i\}$.

1º let $\tau_1 = (t_1, \cdot)$ be the first segment in $\mathbf{T}$, for all CRs c:($\mathbf{S}_c$; t_{c1}, f_{c1}, t_{c2}, f_{c2})$\in\mathbf{C}$ that collide with s_i on τ_1, do 2 º;
  2 ºif $t_{c1} < t_1$, add new CR:($\mathbf{S}_c$; t_{c1}, f_{c1}, t_1, f_{c2}) to $\mathbf{C}$;
3 ºlet $\tau_{-1} = (\cdot, t_2) \in \mathbf{T}$ be the last segment in $\mathbf{T}$, for all CRs c:($\mathbf{S}_c$; t_{c1}, f_{c1}, t_{c2}, f_{c2})$\in\mathbf{C}$ that collide with s_i on τ_2, do 4 º;
  4 ºif $t_{c2} > t_2$, then add new CR:($\mathbf{S}_c$; t_2, f_{c1}, t_{c2}, f_{c2}) to $\mathbf{C}$;
5 ºfor every $\tau = (t_{\tau 1}, t_{\tau 2})$ in $\mathbf{T}$, there are 0, 1, or 2 CRs in $\mathbf{C}$ colliding s_i on τ, do one of 6 º, 7 ºor 8 ºin each case;
  6 ºif no CR in $\mathbf{C}$ collides with s_i on τ, add a new CR:({s}; $t_{\tau 1}$, f_s-δ_f, $t_{\tau 2}$, f_s+δ_f) to $\mathbf{C}$;
  7 ºif one CR:($\mathbf{S}_c$; t_{c1}, f_{c1}, t_{c2}, f_{c2})$\in\mathbf{C}$ collides with s_i on τ, then replace it with CR:($\mathbf{S}_c$+{s_i}; $t_{\tau 1}$, min(f_{c1}, f_s-δ_f), $t_{\tau 2}$, max(f_{c2}, f_s+δ_f));
  8 ºif two CRs:($\mathbf{S}_{c1}$; t_{c11}, f_{c11}, t_{c12}, f_{c12}), ($\mathbf{S}_{21}$; t_{c21}, f_{c21}, t_{c22}, f_{c22})$\in\mathbf{C}$ collide with s_i on τ, then replace them with one CR:($\mathbf{S}_{c1}$+$\mathbf{S}_{c2}$+{s}; $t_{\tau 1}$, min(f_{c11}, f_{c21}), $t_{\tau 2}$, max(f_{c12}, f_{c22})).

Once this update has been performed for all partials of all notes, we have the complete set of collision regions ready. Figure 5 shows the CRs found for the signal in Figure 2. While CRs may overlap themselves, no CR overlaps any sinusoidal partial inside another CR.

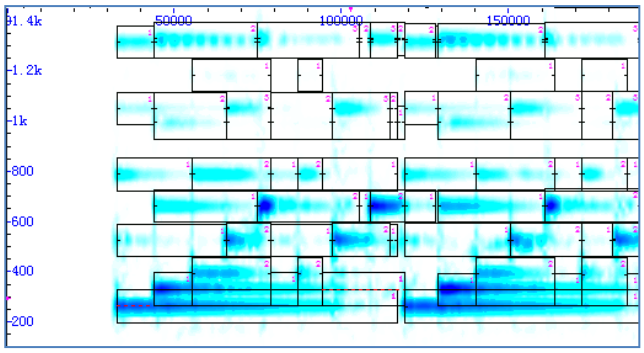


Figure 5 Collision regions identified for Figure 1

6. AMPLITUDE EVALUATION

This section discusses the stationary component estimation stage (stage 3) whose goal, in the current implementation, is to estimate sinusoidal amplitudes based on pre-detected interference details. Estimation of amplitude, like that of frequency, can be corrupted by noise and interference. For the amplitude we adopt

[1] s cannot collide with more than two CRs in $\mathbf{C}$ at any time, because three CRs colliding with s must have collisions among their member sinusoids, so they cannot be all closed.

a similar strategy as we did for frequency, i.e. we measure them directly from data only at "good" atoms. However, since amplitudes are not assumed to satisfy strict constraints as the frequencies do, it is crucial we obtain as many direct estimates as we can to describe amplitudes closely. Assuming that sinusoids within one closed CR do not suffer interference from outside the CR, we do amplitude estimation on a CR-by-CR basis.

6.1. Isolate partials

The simplest CRs are those containing only one sinusoid. Atoms in these CRs are considered free from interference, so can be estimated using any algorithm for estimating solo sinusoid. In this work we use standard orthogonal projection of the spectrum:

$$\lambda = ae^{j\varphi} = \frac{\mathbf{w}^{\mathrm{H}}\mathbf{x}}{\|\mathbf{w}\|^2} \tag{14}$$

where $\mathbf{w}$ is the spectrum of a pure windowed sinusoid at the estimated partial frequency with zero phase, and a and φ are the amplitude and phase angle estimates.

6.2. Least square estimator

Tolonen [15] proposed to use the least square method for estimating "colliding" sinusoids, of which orthogonal projection can be regarded as a special case. For a set of given frequencies, the least square method estimates the amplitudes and phase angles at each frame by solving a linear system involving all the sinusoids. To apply this method we need to know what frequencies are colliding at which frames, which is conveniently handled by collision regions.

For each frame of a closed CR spanning N bins and containing M sinusoids, $N>M$, a linear system of size M is constructed using the spectral data from these N bins. To be more specific, it takes the form of

$$\mathbf{W}^{\mathrm{H}}\mathbf{W}\boldsymbol{\lambda} = \mathbf{W}^{\mathrm{H}}\mathbf{x} \tag{15}$$

$\mathbf{W}$ is an $N\times M$ matrix whose M columns are spectra of pure windowed sinusoids at the M frequencies and zero phase, truncated to contain only the N bins of the CR. $\boldsymbol{\lambda}=(\lambda_1, \ldots, \lambda_M)^{\mathrm{T}}$, $\lambda_m=a_m e^{\mathrm{j}\varphi_m}$, encodes the amplitude and phase angle of the m^{th} sinusoid involved, and $\mathbf{x}$ is the spectrum of this frame, also truncated to those N bins. Noticing that $\mathbf{W}$ remains the same for all frames of a CR, we compute $\mathbf{U}_{M\times N}=[u_{ij}]=(\mathbf{W}^{\mathrm{H}}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{H}}$, then use

$$\boldsymbol{\lambda} = \mathbf{U}\mathbf{x} \tag{16}$$

to compute $\boldsymbol{\lambda}$ for all these frames.

6.3. Exceptions

The least square method assumes that a *good* estimate must yield a *small* square error so as not to produce a large residue. Intuitively, this small square error must be close to the least square error, and ideally we may hope this good estimate also be close to the least square estimate. However, in the note separation task there are two exceptions to such reasoning: that the good estimate may not yield a small square error, and that the proximity of errors may not lead to the proximity of estimates.

6.3.1. Onset exception

When the harmonics of a note is corrupted by an additive noise, a good estimate shall produce a residue close to that noise. In this case it is not adequate to run a least square estimator. A main source of additive noise in a piano recording is note onset attacks. While lasting a very short duration, the onset displays a wide-band behaviour that contaminates the spectrum of harmonics of other on-going notes. Accordingly, at the onset of each note we mark all on-going partials of other notes as not estimable for one frame, so that the least square estimation is not attempted.

6.3.2. Heavy interference exception

The least square estimator usually works fine when $\mathbf{W}$ is well-conditioned. However, as frequencies become close so that the sinusoids get highly correlated, $\mathbf{W}$ gets ill-conditioned and (15) becomes unstable. Intuitively, as the sinusoids become similar to each other, the amplitudes and phase angles can be traded between themselves without incurring much change to the residue, so that a good estimate with nearly least square error may deviate far from the least square estimate. Consequently we may observe large biases on least square estimates of individual sinusoids, even if the total residue is kept minimal.

To evaluate the reliability of amplitude and phase estimated with (16), we consider the sensitivity of the least square estimate of the m^{th} partial with regard to $\mathbf{x}$:

$$J_m = \left(\sum\nolimits_j u_{mj}^2\right)^{1/2} . \tag{17}$$

A change of $\boldsymbol{\delta}$ in the residue can contribute a bias up to $J_m\cdot\|\boldsymbol{\delta}\|$ to λ_m. The smaller is J_m, the more likely has λ_m a reliable estimate. In a CR which contains only one sinusoid, $J_1=(1+\varepsilon_w)\ \|\mathbf{w}\|^{-1}$, where $\|\mathbf{w}\|$ is the L^2 norm of the window spectrum, and $|\varepsilon_w|<<1$ is attributed to the truncations used to construct $\mathbf{W}$. We use $J_m\|\mathbf{w}\|$ as an indicator of the reliability of least square estimates. Amplitudes are estimated with (16) for partial m only if $J_m\|\mathbf{w}\|$ is below a preset threshold, e.g. 2. Other amplitudes are marked not estimable and passed on to the next stage (6.4).

6.4. Interpolation and extrapolation

Amplitude parameters that have not been measured due to reliability concerns are inferred from the already estimated ones by means of interpolation and extrapolation. Using such "guessing" techniques means we no longer target accurate additive separation as an objective. However, by common-sense design we may still maintain the naturalness of separated notes and their perceptual resemblance to what is heard in the original recording.

6.4.1. Interpolation

Interpolation is applied on a partial-by-partial basis to atoms whose amplitudes have not been estimated but lie between other atoms whose amplitudes have. More specifically, if amplitudes have been measured for a partial m at frames l_1 and l_2, $l_1<l_2$-1, but not at frames between the two frames, then we interpolate between the two frames exponentially:

$$a_{m,l} = \left(a_{m,l_1}^{l_2-l} a_{m,l_2}^{l-l_1}\right)^{\frac{1}{l_2-l_1}}, \ l=l_1+1, \ldots, l_2\text{-}1. \tag{18}$$

where $a_{m,l}$ is the amplitude of partial m at frame l.

The interpolation stage fills the gaps between atoms with measured amplitudes, but does not estimate amplitudes of atoms at the start and end of partials. For these an extrapolation stage is involved.

6.4.2. Extrapolation using amplitude modulation profile

To infer amplitudes before the first or after the last direct estimates of a sinusoid, we need to make assumptions about amplitude laws in these places. While it is always possible to extrapolate directly from the measured amplitudes of each partial, doing so, according to our experiments, is not advisable near onsets or far beyond measured atoms. On the other hand, since various partials can have reliable estimates during different intervals, it is possible to make cross-partial reference. In this paper we compute an *amplitude modulation (AM) profile* from the already estimated amplitudes for this purpose.

An AM profile is a function of time that describes the overall amplitude variation rate of the partials of a note. Let l be the frame index, the AM profile, denoted by P_l, is given as

$$P_l = \frac{\sum_{m=1}^{M} a_m \log \frac{a_{m,l}}{a_{m,l-1}}}{\sum_{m=1}^{M} a_m}, \; l=1, \ldots, L\text{-}1. \tag{19}$$

where L is the length of the note in frames, $a_{m,l}$ is the amplitude of partial m at frame l, and $a_m = \sum_l a_{m,l}$. P_l is interpreted as the average of amplitude rates at frame l weighted by partial amplitudes.

If none of the partials of a note has reliable amplitude estimates for the first L_1 frames (onset frames), we cannot compute P_l with (19) at frames 1 to L_1. In this case we obtain rough amplitude estimates by orthogonal projection[2], assuming partials near the onset have dominating energy. These rough amplitudes are used to estimate P_l for l=1, …, L_1-1. P_{L_1} is linearly interpolated from P_{L_1-1} and P_{L_1+1}.

If none of the partials of a note have reliable estimates for the *last* L_2 frames, we cannot turn to orthogonal projection like for onset frames, as leftover partials near offsets are often masked by noise or interference. For these frames we simply linearly extrapolate the AM profile itself, while taking special care that the amplitude rate be non-positive.

Further smoothing can be applied to the AM profile for improved smoothness. Once the AM profile is ready the extrapolation is done by applying the profile directly:

$$a_{m,l} = e^{P_l} \cdot a_{m,l-1} \text{ (forward)} \tag{20a}$$

$$a_{m,l} = e^{-P_{l+1}} \cdot a_{m,l+1} \text{ (backward)} \tag{20b}$$

7. TRANSIENTS

This section discusses the transient extraction stage (stage 4), which separates the initial attack of each note from audio. Piano notes have transients at keystrokes. Although in theory all sounds can be represented as sinusoids, the sinusoidal representation of a transient would involve faster amplitude and frequency changes than the standard technique could handle, and the physical meanings of the parameters would be unclear.

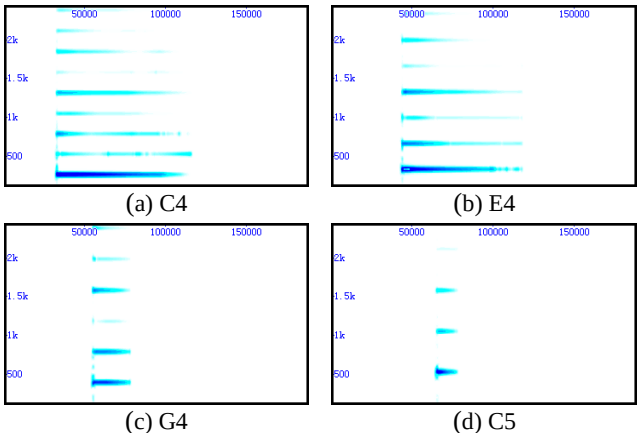
In this work we simply represent the transient with the complex spectrum. The transient is assumed to stretch the length of one long "transient" frame before the start of the harmonics. Let the DFT of this frame be $X(k)$, k=0, …, K/2-1, where K is the length of the transient frame; let f^1, …, f^M be the frequencies of the note(s) starting with this transient, and g^1, …, g^N be the frequencies of other on-going sinusoids during this frame. We derive the spectrum of the transient by notching out the on-going sinusoids, while preserving the starting ones:

1 °for each g^n, n=1, …, N, remove 4 bins from $X(k)$ centred at g^nK by setting at value at these bins to 0;
2 °for each f^m, m=1, …, M, recover 4 bins centred at f^mK by restoring these bins to their original value.

The separated transient is synthesized from the spectrum with inverse DFT. To reconstruct a complete note, we join the transient to the harmonics with standard overlap-add method. We initialize the phase angles of the harmonics to their spectral phases at the first frame. As the same phases are also preserved in the transient, the transition between transient and harmonics is kept smooth.

8. RESULT

We run our note separator on a commercial recording of Bach's Prelude in C, BWV 846a, using one channel sampled at 44.1kHz. The initial part of its spectrogram is given in Figure 2. Figure 6 shows the separation results for the first four notes. Graphically these single-note spectrograms look smoother than the originals in Figure 2, owing to the explicit modelling as constant-frequency harmonic sinusoids. We do observe irregularities in the harmonics where they used to overlap, but the result is well contained in plausible range. Since guessed parameters have guaranteed smoothness, we know that these less regular parts are the result of direct estimates from data, and it is these parts that pinned down the characteristics of the notes in the original recording. Perceptually we have found no audible artefact with the separated notes, and heard very little difference between the separated notes and their counterparts in the original recording.



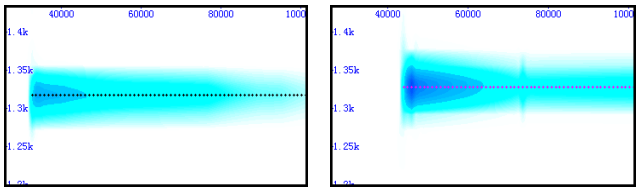
(a) C4 (b) E4

(c) G4 (d) C5

Figure 6 First four notes separated

Figure 7 shows details of Figure 6(a) and Figure 6(b) around 1.3kHz, i.e. the 5th partial of the first note (C4) and the 4th partial of the second note (E4). Although these two partials were shown to interfere heavily in the original spectrogram (Figure 3), our method is able to obtain reasonably clean separation from the same spectrogram. A small anomaly in the result is the inaudible amplitude spike estimated for the second note when the fifth note (E5) kicks in, showing that our handling of parameter estimation during other notes' onsets still has room for improvement.

[2] Or any other estimator for single, interference-free sinusoids.



(a) C4: 5th partial (b) E4: 4th partial

Figure 7 Details of Figure 6

Figure 8 shows the reconstruction we get by summing up all resynthesized notes in Figure 1. As with separated notes, the reconstructed spectrogram has a smoother and sharper look than the original. Because of parametric modelling, much of the non-music content in the original recording, such as background noise, the pianist's humming-along, and other unidentified extras, are removed from the reconstruction. Listening comparison between the reconstruction and the original shows the two highly similar to the ear. Minor difference can be heard at some note attacks, mainly towards the end of the recording where low-pitched notes become more frequent. Since low pitches introduce more interference to the harmonics and eats out more spectral bins from the transient, we can expect some performance loss on both the stationary and the transient parts.

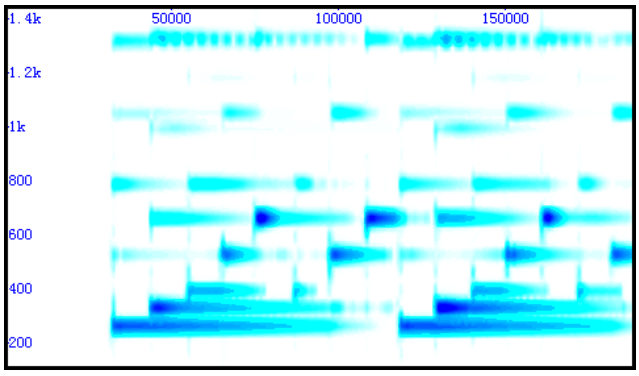


Figure 8 Reconstruction of recording

9. SUMMARY AND FUTURE WORK

In this paper we have presented a semi-automatic system for separating a piano recording into individual notes. The system employs two concepts: the imitation method that allows us to trade between modelling accuracy and result stability, and the collision region that helps organize systematic handling of interfering sinusoids. Our results show that the system is able to obtain separated notes with high quality and true to the original recording even though the signal processing techniques involved at fairly basic. The notes are represented with a transient-plus-harmonics model, therefore are ready to be used by all algorithms handling sinusoidal models, in addition to those handling monophonic or general audio waveforms. Direct reconstruction from the separated notes gives a de-noised version of the recording.

While reasonably self-contained for proof of concept, the proposed system is only a small step towards general note separation task. The system can be improved in many ways for sound quality, robustness and usability. For example, the phase angle at individual frames may be reincorporated during or after amplitude estimation to preserve minor pitch and wave shape variations. For amplitude estimation we may use multiple window sizes up to the duration of the collision region instead of a constant size of 2048. The transient energy where it collides with ongoing sinusoids may be partially restored, and the handling of transient may go beyond one frame to capture the full force and span of note attacks. We may also start bringing musical instruments with continuously variable pitches, as well as non-pitched instruments, into the picture.

10. ACKNOWLEDGEMENT

Part of this work was finished when the author was with Queen Mary, University of London and Professor Mark Sandler, with support from the Centre for Digital Music platform grant.

11. REFERENCES

[1] S. Ewert, B. Pardo, M. Müller and M.D. Plumbley, "Score-informed source separation for musical audio recordings: an overview," *IEEE Signal Processing Magazine*, vol.31 no.3, pp.116-124, May 2014.

[2] R. Hennequin, B. David and R. Badeau, "Score informed audio source separation using a parametric model of non-negative spectrogram," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Prague, 2011, pp.45-48.

[3] N. Bryan and G. Mysore, "Interactive refinement of supervised and semi-supervised sound source separation estimates," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vancouver, 2013.

[4] P.-K. Jao, Y.-H. Yang and B. Wohlberg, "Informed monaural source separation of music based on convolutional sparse coding," in Proc. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Brisbane, 2015.

[5] S. N. Levine and J. O. Smith III, "A sines+transients+noise audio representation for data compression and time/pitch scale modifications," in *Proc. AES 105th Convention*, San Francisco, 1998.

[6] N.H. Fletcher and T.D. Rossing, *The Physics of Musical Instruments* (2nd ed.), Springer-Verlag New York Inc., 1998.

[7] A. Gunnarsson & I. Gu, "Music signal synthesis using sinusoid models and sliding-window ESPRIT," in *Proc. IEEE International Conference on Multimedia and Expo*, Toronto, 2006.

[8] J. P. Bello, L. Daudet and M. B. Sandler, "Automatic piano transcription using frequency and time-domain information," *IEEE Transactions on Audio, Speech, and Language Processing*, vol.14, no.6, 2006, pp.2242-2251.

[9] S. Sigtia, E. Benetos and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol.24, no.5, 2016, pp.927-939.

[10] K. O'Hanlon and M. D. Plumbley, "Polyphonic piano transcription using non-negative matrix factorisation and group sparsity," in Proc. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Florence, 2014.

[11] M. Marolt, "A connectionist approach to automatic transcription of polyphonic piano music," *IEEE Transactions on Multimedia*, vol.6 no.3, 2004, pp.439-449.

[12] R.J. McAulay and T.F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol.34, no.4, 1986, pp.744-754.

[13] X. Serra, "Musical sound modeling with sinusoids plus noise," *Musical Signal Processing*, Swets & Zeitlinger Publishers, 1997.

[14] Wen X. and M. Sandler, "Sinusoid modeling in a harmonic context," in *Proc. 10th International Conference on Digital Audio Effects (DAFx)*, Bordeaux, 2007.

[15] T. Tolonen, "Methods for separation of harmonic sound sources using sinusoidal modeling," in *Proc. AES 106th Convention*, Munich, 1999.