SoundMathWorks

Technical Report: spectral analysis of music

William J. DeMeo

SoundMathWorks

Technical Report: spectral analysis of music 23 July 2001

William J. DeMeo
william.demeo@verizon.net

Contents

Abstract

This paper describes new and existing methods for representing, analyzing, and modifying the information content of musical signals. In particular, the primary objective is to find signal representations well suited for the analysis and understanding of the human perception of music. In pursuit of this objective, we consider how to use such representations to make inferences about a listener's musical preferences concerning the represented signal. We also consider ways in which special signal representations can facilitate signal modifications accroding to given musical criteria.

1 Introduction

1.1 Analysis of Non-stationary Signals

There is a vast literature on the analysis of signals – be they musical or not. The broad class of analysis methods that we study here consists of *time-frequency representations*. The type of signal which concerns us is a *time series*, $x(t)$. In other words, it is a function of a single time variable, t. Often there is reason to believe that periodicities play an important role in the structure of $x(t)$. To study the signal from this point of view, we decompose it as a function of both time and frequency. Such a decomposition is what we mean by a time-frequency representation (TFR).

The two broad classes of TFR's which we consider are *atomic decompositions* and *energy distributions*. The first decomposes a signal by "projecting" it onto the time-frequency space, thereby equating it with a weighted sum of basic elements in that space. (These basic elements do not always comprise a *basis* – a full set of orthonormal elements – and, in such cases, the decomposition is not a true projection.) A few more comments about atomic decompositions appear below, but a more detailed discussion appears is section 3.

The second class of TFR's are the energy distributions, the primary objective of which is to represent a distribution of the signal's energy across the time-frequency plane. We consider the energy distributions in section 4.

Before proceeding, let's consider some general, motivating comments concerning the use of atomic decompositions for our application. The primary consideration for such analyses is the selection an appropriate basis for processing a particular class of signals. The decomposition coefficients of a signal in a basis define a representation that highlights some particular signal properties. For example, wavelet coefficients provide explicit information on the location and type of signal singularities. The problem is to find a criterion for selecting a basis that is intrinsically well adapted to represent a class of signals.

Using classical Fourier analysis, a signal x can be decomposed as a weighted sum of sinusoidal functions, $\{e^{i2\pi\xi t}\}$,

$$x(t) = \int_{-\infty}^{\infty} X(\xi) e^{i2\pi\xi t}\, d\xi$$

with weights (amplitudes) given by the Fourier transform of x:

$$X(\xi) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi\xi t}\, dt \tag{1}$$

If our focus is limited to stationary signals whose frequency contents do not change dramatically with time, the Fourier analysis provides simple answers to most questions. However, we are interested in representing transient phenomena – e.g. a sudden change or singularity in a musical signal – and the Fourier transform is the wrong tool for the job.

The Fourier coefficient in (1) is obtained by correlating x with a sinusoidal wave $e^{i2\pi\xi t}$. Since the support of $e^{i2\pi\xi t}$ covers the whole real line, $X(\xi)$ depends on the values $x(t)$ for all times $t \in \mathbb{R}$. This global "mix" of information makes it difficult to analyze any local property of x from X. We need to consider using local time-frequency transforms, such as the windowed Fourier transform and wavelet transform, which decompose the signal over waveforms that are well localized in both frequency and time.

Though wavelet and windowed Fourier techniques can be effectively applied to non-stationary signals, they suffer from the adverse consequences of Heisenberg's uncertainty principle. In TFR

analyses, the manifestation of this principle is the time-frequency resolution trade-off which is described in section 3.3. Addressing this problem are the more general distributions of section 4.

1.2 Signal Processing for Perception Analysis

A second objective of this work is to derive new methods and software routines for a *perception analysis* of a piece of music, $x(t)$. We consider the aforementioned signal analysis methods in light of this objective, and focus on those methods which facilitate measures of perceived qualities of music. More specifically, we consider how signal energy densities relate to a quantitative measure of *sensory dissonance*, which we define in section 2.3.

There are great challenges to overcome even when we limit our scope to the spectral analysis outlined in the preceding section. By further stipulating that this will then serve as a foundation for the analysis of perceptions of musical signals, we significantly increase the problem's complexity. Thus, it is helpful to consider what we should take to be the primary tasks involved in reaching our objective. The following is a rough description of these tasks:

1. Find useful time-frequency representations for analyzing the information content of $x(t)$, with the goal of characterizing perceptual properties of the signal.

2. Exploiting the analyses of 1, derive measures of qualities related to human perception of the signal. Derive the "consonance signature" of $x(t)$.

3. Create methods for altering the consonance signature of $x(t)$, by manipulating the structure of its time-frequency representation (e.g. using time-frequency transformations).

Our approach to (1) considers the atomic decompositions (e.g. windowed Fourier or wavelet analysis), as well as energy distributions. Concerning the former, the "grains" of *granular synthesis* (Cavalierre and Piccialli [1], [2]) usually represent segments of the signal that extend beyond a single period. The goal is to capture the pseudo-periodic component of the signal. For wavelet analyses, we could attempt to find a basis whose elements are closely correlated with the components of the signal having the greatest "periodic endurance." We decompose the signal into its quasi-harmonic part (wavelet components) and its *transients* (error terms, deviations, micro-variants):

$$x(t) = \mathrm{P}_{\mathcal{M}}[x](t) + [x(t) - \mathrm{P}_{\mathcal{M}}[x](t)]$$

The first term on the right is the orthogonal projection of x onto the "harmonic" subspace $\mathcal{M}$. The second term accounts for the part of x that is orthogonal to the harmonic subspace (i.e. the transients).

To address item (2), we might consider various decompositions of frequency space F, for instance, as a direct sum of subspaces $\{S_1, S_2, \ldots, S_N\}$:

$$F = S_1 \oplus S_2 \oplus \cdots \oplus S_N$$

where each S_k defines a *subspace of great consonance.* This could be taken to mean that tones occurring within the same subspace, say $x \in S_k$ and $x' \in S_k$, are relatively consonant, while x and $y \in S_{k+1}$ are relatively dissonant. We could also specify some properties that such subspaces should posses. For example, let S_k and S_{k+1} be two distinct subspaces of great consonance and suppose $x \in S_k$ and $y \in S_{k+1}$. Then,

$$D(x, x') \leq D(x, y) \text{ for any } x' \in S_k$$

More precisely, this is a property of the *dissonance metric* $D(\cdot, \cdot)$ defined on the space, which leads us to the issue of how dissonance ought to be defined.

The concept of *sensory dissonance* was originally proposed by Helmholtz [3], and further developed by Plomp and Levelt [4], and Sethares [5]. Though we consider these studies in greater detail in sections 2.3 and 2.4, what follows is a descriptions that motivates our alternative treatment of this subject.

In order to assess the intrinsic dissonance of a musical signal over a small time interval, the aforementioned studies employ a function of the the signal's estimated frequency components over that interval. This often provides a useful quantitative measure. However, such a function makes no attempt to account for other widely accepted notions of dissonance. Perhaps the most obvious short-coming results from the point-wise nature of the dissonance function. That is, because it is well localized in time, the dissonance function does not measure the *melodic dissonance* of the signal. The melodic dissonance of a given segment of music depends on that segment's relation to its context. In our present work, we consider signal analysis methods that provide for more dynamic dissonance measures, simultaneously accounting for local dissonance as well as melodic contour.

2 Background: Musical Signals

2.1 Mathematics of the Pure Tone

The pure tone is the most elementary of all signals (Hartmann [6]). Mathematically it is known as the sine wave, a function of time described by the equation

$$x(t) = A\sin(2\pi t/T + \phi) \tag{2}$$

where A is the *amplitude*, T is the *period* in seconds, and ϕ is the *phase* in radians. The units of x, whatever they might be (mechanical displacement, pressure, voltage), are the same as the units of the amplitude A.

The sine function is periodic; it repeats itself when its argument, the *instantaneous phase* $2\pi t/T + \phi$, changes by 2π. Equation (2) shows that if t starts at t_0 and increases to $t_0 + T$, the function comes back to its starting point; that is,

$$x(t_0) = x(t_0 + T)$$

This is why T is called the period, measured in units of seconds per cycle. Its inverse, measured in units of cycles per second (Hz) is called the *frequency* of the wave,

$$f = 1/T$$

Because there are 2π radians in a cycle, the *angular frequency*, ω, is related to the frequency by

$$\omega = 2\pi f$$

The angular frequency is measure in radians per second. Therefore, one can write the sine wave with respect to these alternative units of frequency:

$$x(t) = A\sin(2\pi f t + \phi) = A\sin(\omega t + \phi)$$

Like any other periodic tone, the pure tone can be characterized by the psychological dimensions of *pitch, loudness,* and *tone color*. The pure tone actually serves as a reference for pitch because of the latter's relation to the physical quantity of frequency.

The standard textbook range of audible frequencies is from 20 Hz to 20000 Hz (Hartmann *op. cit.*). The upper limit depends greatly on age and otological history. For the majority of young adults the upper limit is likely to be closer to 17,000 Hz. The lower limit of 20 Hz is also problematic. In order to make a 20-Hz tone audible one must use a strong signal, but this runs the risk of creating harmonic distortion, particularly at the third harmonic (60 Hz), because the hearing organ is so much more sensitive at 60 than at 20 Hz.

The term *tone color* refers to that part of timbre that is attributable to the steady state part of a tone; i.e., the tone without transients associated with onset, offset, or ongoing aperiodic fluctuations (Hartmann *op. cit.*). A pure tone with a frequency below 200 Hz is judged as "dull," while a tone with a frequency greater than 2000 Hz is "bright." Most of the energy in speech signals is below 2000 Hz, and although numerous medieval woodwind instruments had spectra that extended to high frequencies, most modern musical instruments act as low-pass filters that attenuate harmonics with frequencies greater than 1000 Hz. This is true of strings, brass, and woodwind instruments. There are exceptions – the piccolo, for example – and there are organ pipes with fundamental frequencies as high as 8372 Hz (C_9).

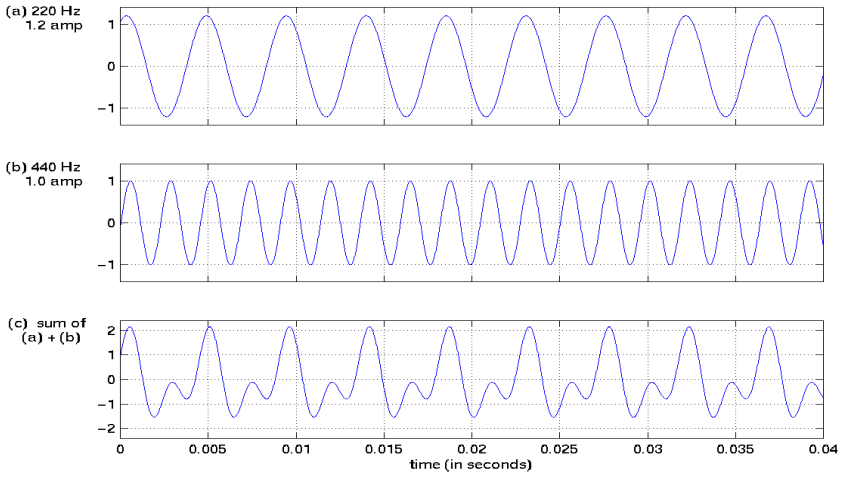

Figure 1: Synthesis by addition of two pure sine waves, (a)+(b)=(c).

2.2 Analysis and Synthesis

Although it can produce a tone, an individual sine wave has little musical value by itself. However, combinations of sine waves can be used to describe, analyze, and synthesize almost any possible sound. A complex sound can be modeled as a function of time, $x(t)$. It can be decomposed into a family of simple sine waves, each of which is characterized by its frequency, amplitude, and phase. These individual waves are called the *partials* (or *overtones*) and the collection of all the partials is called the *spectrum* of x. If the overtones occur at integer multiples of the lowest frequency (fundamental) wave, then the overtones are sometimes called *harmonics*.

As a simple example, consider a sine wave with unit amplitude and a frequency of 440 Hz – so called "concert A", or A_4. Then introduce a second sine wave with amplitude 1.2 and frequency 220 Hz (one *octave* below the first). These two waves are shown as (a) and (b) in Figure 1. When the waves are sounded together, the amplitudes are added together point by point and the result is the complex wave shown in (c) of the figure. This example illustrates the distinction between what is known as a "pure tone," (a) and (b), and a "complex tone," (c).

The technique of expressing a function as a sum of sinusoidal components has been around for at least 200 years. It is important not only for our analytic understanding of sound waves, but also for understanding the human perception of sounds. It is believed that the ear functions as a kind of biological spectrum analyzer. That is, when sound waves impinge on the ear, what we perceive is a direct result of the spectral components of the sound, and is only indirectly a result of the composite waveform.

Take the wave forms in Figure 1 for example. Our eardrum vibrates under the influence of the composite wave appearing in part (c). However, the mechanisms of the inner ear – our spectrum analyzer – enable us to perceive the sound more precisely as a composition of (a) and (b). In Figure 2, sine wave (b) is still modulating at 440 Hz, but it now begins at a different phase. Therefore, the composite wave form shown in part (c) of Figure 2, has a very different shape than that of Figure 1. Yet the human ear perceives the same sound, since both waves comprise the same frequency components.

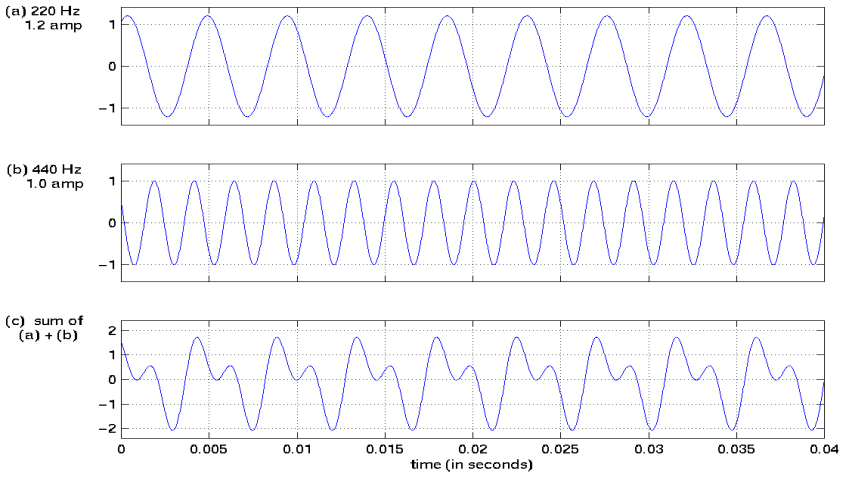

Figure 2: Synthesis by adding sine waves having the same frequencies as those of Figure 1, but now the phase of sine wave (b) has been shifted. Though waveform (c) looks different from that of Figure 1, the same sound is perceived.

The foregoing presents a very basic idea underlying the formation of the complex tone, $x(t)$. It can be expressed mathematically as

$$x(t) = \sum_{k=1}^{K} a_k \cos(\omega_k t + p_k) \tag{3}$$

where the a_k define the amplitudes associated with each partial and the p_k are some (usually arbitrarily specified) phases. For example, the figures above represent the function

$$x(t) = 1.2\cos(220t + p_1) + 1.0\cos(440t + p_2)$$

for $0 < t \leq (0.04)2\pi$ and $0 < p_k \leq 2\pi$.

The relevance of expression (3) for the analysis of music is limited by the fact that real world signals, unlike those in the figures, are never exactly periodic. Strong similarities among segments of signals often occur, which give rise to a sense of pseudo-periodicity and, hence, pitch. However, superimposed on this periodicity are fluctuations, noise, small frequency variations, etc. These aperiodic components are due to different phenomena connected with the particular physical mechanism used for the production of the sound under analysis (Cavaliere and Piccialli). For this reason, a realistic model of a signal must generalize this simple, time-invariant addition of periodic sine waves. It should allow both the amplitudes and frequencies to vary with time. Writing $a_k = a_k(t)$ and $\omega_k t + p_k = \phi_k(t)$ accomplishes this and yields the following sum of partials:

$$x[t|\beta(t)] = \sum_{k=1}^{K} a_k(t) \cos(\phi_k(t)) \tag{4}$$

where the notation $x(\cdot|\beta)$ indicates that the functional form of x is conditional on the time-varying vector $\beta(t) \equiv (a_1(t), \ldots, a_K(t), \phi_1(t), \ldots, \phi_K(t))^t$ of amplitudes and phases. This model also requires that $a_k(t)$ and $\phi_k'(t)$ vary slowly [7]. This requirement as well as the relationship between the

"instantaneous frequency," ω_k, and the time-varying phase, $\phi_k(t)$, will be discussed in the following sections.

Still implicit in model (4) is the assumption that the signal is well approximated by a sum of pure sinusoidal waves. However, the presence of a given frequency, ω_k, can now be short lived, and its contribution to x varies (albeit slowly) with time via $a_k(t)$.

Assuming model (4), the problem of analyzing a signal becomes the estimation of $\beta(t)$, and there are as many approaches to that problem as there are applications of signal analysis.

In later sections, we consider which signal analysis methods are particularly well suited to the type of musical analysis we wish to perform. Therefore, it is crucial that we have a good grasp of the musical ideas underlying and motivating our research into musical signal analysis. The following section presents one such notion – musical *consonance*. Also presented are some existing quantitative measures of this concept.

2.3 Consonance and Dissonance

According to Tenney [8] and Sethares [5], the historical usage of the term *consonance* (and its antonym, *dissonance*) can be classified according to five distinct categories. (see [5] for more details)

1. **Melodic Consonance (CDC-1)** successive melodic intervals are consonance or dissonant depending on the surrounding melodic context; refers to relatedness of pitches sounded successively (melodic contour)

2. **Polyphonic Consonance (CDC-2)** consonance is a function of the interval between (usually 2) simultaneously sounding tones; proponents of this definition are clearest about the consonant $\leftrightarrow$ "pleasant" association.

3. **Contrapuntal Consonance (CDC-3)** consonance is defined by its role in counterpoint, e.g. voice-leading techniques; the 4th is declared dissonant (in contrast to CDC-2).

4. **Functional Consonance (CDC-4)** focus on relation of individual tones to the root or tonic; consonant tones are those that are in simple unison, third, or fifth relation to root. Dissonance occurs when the music moves away from root and sets up a desired return; dissonances cause chordal motion (and not vice-versa); Piston in "Harmony" relates consonant with stable or complete and dissonant with restless or requiring resolution; "the essential quality of dissonance is its sense of movement and not...its degree of unpleasantness to the ear."

5. **Sensory Consonance (CDC-5)** equates consonance with smoothness and the absence of beats; equates dissonance with roughness and the presence of beats.

The definition of sensory consonance is based on the phenomenon of beats. If two pure sine tones are sounded at almost the same frequency, then beating occurs due to the interference between the tones. The beating becomes slower as the two frequencies approach each other and disappears when they coincide. Typically, slower beats are perceived as gentle and pleasant while fast beats are perceived as rough and unpleasant. Observing that any sound can be decomposed into sinusoidal partials, Helmholtz [3] theorized that the perception of dissonance in a musical tone is determined by the presence and quality of beats among the tone's interacting partials.

The present research effort is directed at the discovery of a definition that might reconcile these competing notions of consonance. In particular, we would like to exploit the theory of *sensory* and *tonal* consonance (CDC-2 and CDC-5) of Plomp and Levelt [4] as well as its elaboration in [5]. Some of this theory is described below in section 2.4. Briefly, it employs functions called *dissonance curves* which measure the "sensory" dissonance, of a complex tone at each particular instant in time.[2] This provides a useful point-wise measure. However, we would also like a measure that is dynamic and appeals to a melodic sense of consonance. For example, a dissonance curve does not account for dissonance due to melodic changes from one complex tone to the next.

2.4 Dissonance Curves

Two pure sine waves are both perceptible if the frequencies are well separated. When the frequencies are close together, only one sine wave is heard (though possibly with beats). By varying the frequency of one of the waves while keeping the other wave fixed, Plomp and Levelt [4] gathered data revealing subject's preferences for various frequency intervals. An example of what these data indicate appears as a *dissonance curve* in Figure 3. (One method of deriving such curves in practice is described in section 5.2.) The curves in the figure are in stark contrast to existing musical theory. In particular they imply that, in terms of sensory dissonance, intervals such as the major 7th and minor 9th are almost indistinguishable from the octave. This leads some to argue that these data bear no relation to our notions of musical consonance. Regardless of whether they have a direct influence on our ideas about musical intervals, dissonance curves provide a way to measure the intervals among *pure sine waves*. Therefore, they can be useful when analyzing a musical signal that has been decomposed into its spectral components.

[2] Really, a small interval of time is required to ascertain what (pseudo-)periodic frequencies are present at a particular instant.

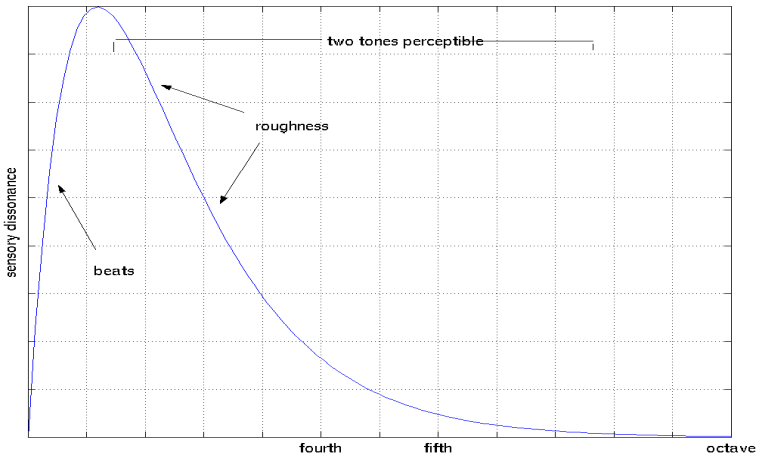

Figure 3: Two sine waves are sounded simultaneously. Typical perceptions include pleasant beating (at small frequency ratios), roughness (at middle ratios), and separation into two tones (at first with roughness, and later without) for larger ratios. the horizontal axis represents the frequency interval between the two sine waves, and the vertical axis is a normalized measure of "sensory" dissonance. The frequency of the lower sine wave is 400 Hz

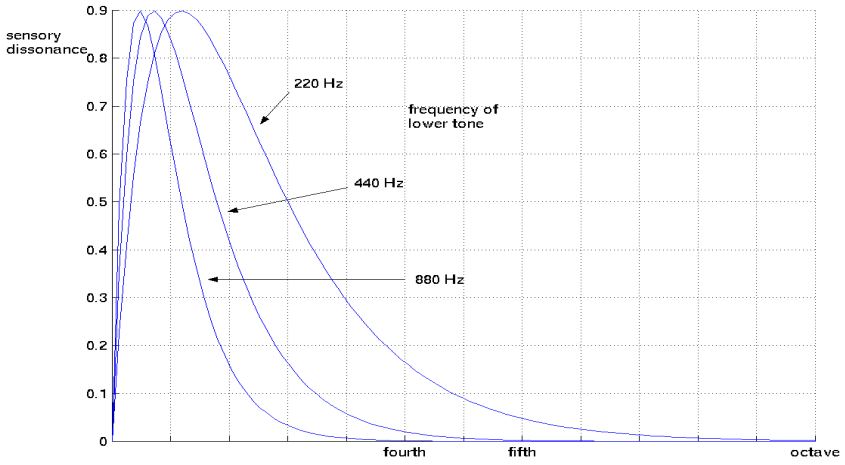

Figure 4: Two sine waves are sounded simultaneously. The horizontal axis represents the frequency interval between the two sine waves, and the vertical axis is a normalized measure of "sensory" dissonance. The plot shows how the sensory consonance and dissonance change depending on the frequency of the lower tone.

3 Atomic Decompositions

3.1 Time-Frequency Atoms

A linear time-frequency transform correlates the signal x with a family of waveforms that are well concentrated in time and in frequency. These waveforms are called *time-frequency atoms*. Let us consider a general family of time-frequency atoms $\{\phi_\gamma\}_{\gamma\in\Gamma}$, where γ might be a multi-index parameter. We suppose that $\phi_\gamma \in L^2(\mathbb{R})$ and that $\|\phi_\gamma\| = 1$. The corresponding linear time-frequency transform of $x \in L^2(\mathbb{R})$ is defined by

$$\begin{aligned} \mathrm{T}[x](\gamma) = \langle x, \phi_\gamma \rangle &= \int_{-\infty}^{\infty} x(t)\phi_\gamma^*(t)\,dt \\ &= \langle X, \Phi_\gamma \rangle = \int_{-\infty}^{\infty} X(\omega)\Phi_\gamma^*(\omega)\,d\omega \end{aligned} \tag{5}$$

The third equality holds by the Parseval formula (A.4).

If $\phi_\gamma(t)$ is approximately zero when t is outside a neighborhood of an abscissa u, then $\langle x, \phi_\gamma \rangle$ depends only on the values of $x(t)$ in this neighborhood. Similarly, if $\Phi_\gamma(\omega)$ is negligible for ω outside a neighborhood of ξ, then (5) shows that $\langle x, \phi_\gamma \rangle$ characterizes $X(\omega)$ in that neighborhood.

Energy Density. Suppose that for any point (u, ξ) in the time-frequency plane there exists a unique atom $\phi_{\gamma(u,\xi)}$ centered at this point. The time-frequency support of $\phi_{\gamma(u,\xi)}$ specifies a neighborhood of (u, ξ) where the energy of x is measured by

$$\begin{aligned} E_{\mathrm{T}[x]}(u, \xi) &= |\mathrm{T}[x](\gamma)|^2 \\ &= \left|\langle x, \phi_{\gamma(u,\xi)} \rangle\right|^2 \end{aligned} \tag{6}$$

Later we will see that any such energy density is an averaging of the *Winger-Ville transform*, with a kernel that depends on the atoms ϕ_γ.

3.2 Windowed Fourier Transform

In 1946 Gabor [9] introduced windowed Fourier atoms to measure the "frequency variations" of sounds. A real and symmetric window $g(t) = g(-t)$ is translated by u and modulated by the frequency ξ:

$$g_{u,\xi}(t) = e^{i2\pi\xi t} g(t - u)$$

The original window, g, has unit norm and therefore $\|g_{u,\xi}\| = 1$ for any $(u, \xi) \in \mathbb{R}^2$. The resulting windowed Fourier transform of $x \in L^2(\mathbb{R})$ is

$$\begin{aligned} \mathrm{S}[x](u, \xi) &= \langle x(t), g_{u,\xi} \rangle \\ &= \int_{-\infty}^{\infty} x(t) g(t - u) e^{-i2\pi\xi t}\,dt \end{aligned} \tag{7}$$

This transform is also called the *short time Fourier transform* because the multiplication by $g(t-u)$ localizes the Fourier integral in the neighborhood of $t = u$.

As in (6), one can define an energy density, called a *spectrogram*, by

$$\begin{aligned} E_{\mathrm{S}[x]}(u,\xi) &= |\mathrm{S}[x](u,\xi)|^2 \\ &= |\langle x(t), g_{u,\xi}\rangle|^2 \end{aligned}$$

The spectrogram measures the energy of x in the time-frequency neighborhood of (u,ξ) specified by the Heisenberg box of $g_{u,\xi}$.

3.3 Wavelet Transforms

To analyze signal structures of very different sizes, it is necessary to use time frequency atoms with different time supports. The wavelet transform decomposes signals over dilated and translated wavelets. A wavelet is a function $\psi \in L^2(\mathbb{R})$ with zero average:

$$\int_{-\infty}^{\infty} \psi(t)dt = 0$$

It is normalized, $\|\psi\| = 1$ and centered in the neighborhood $t = 0$. A family of time-frequency atoms is obtained by scaling ψ by s and translating it by u:

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right)$$

These atoms remain normalized: $\|\psi_{u,s}\| = 1$. The wavelet transform of a function $x \in L^2(\mathbb{R})$ at time u and scale s is:

$$\mathrm{W}[x](u,s) = \langle x, \psi_{u,s}\rangle = \int_{-\infty}^{\infty} x(t)\frac{1}{\sqrt{s}}\psi^*\left(\frac{t-u}{s}\right)dt \tag{8}$$

Like a windowed Fourier transform, a wavelet transform can measure the time evolution of frequency transients. This requires using a complex analytic wavelet, which can separate amplitude and phase components. An analytic wavelet can be used to measure *instantaneous frequencies*. In contrast, real wavelets are often used to detect sharp signal transitions.

Analytic Signals. To analyze the time evolution of frequency tones, it is necessary to use an analytic wavelet to separate the phase and amplitude information of signals. A function $x_a \in L^2(\mathbb{R})$ is said to be *analytic* if its Fourier transform is zero for negative frequencies:

$$X_a(\omega) = 0 \text{ if } \omega < 0.$$

An analytic function is necessarily complex but is entirely characterized by its real part. Indeed, the Fourier transform of its real part $x = \mathrm{Re}[x_a]$ is

$$X(\omega) = \frac{X_a(\omega) + X_a^*(-\omega)}{2}$$

and this relation can be inverted to yield

$$X_a(\omega) = \begin{cases} 2X(\omega) & \text{if } \omega \geq 0 \\ 0 & \text{if } \omega < 0 \end{cases} \tag{9}$$

The analytic part $x_a(t)$ of a signal $x(t)$ is the inverse Fourier transform of $X_a(\omega)$ defined by (9).

Equivalently, we obtain the analytic part of $x(t)$ a signal by operating on the signal with the *Hilbert transform*. Precisely, given a real signal $x(t)$, the analytic part is

$$x_a(t) = x(t) + i\,\mathrm{H}[x](t) = x(t) + \frac{i}{\pi}\,\mathrm{pv}\int_{-\infty}^{\infty}\frac{x(s)}{t-s}ds$$

where H denotes the Hilbert transform.

Time-Frequency Resolution. An analytic wavelet transform is calculated with an analytic wavelet ψ with the integral (8). Its time-frequency resolution depends on the time-frequency spread of the wavelet atoms $\psi_{u,s}$.

We suppose that ψ is centered at 0, which implies that $\psi_{u,s}$ is centered at $t = u$. As shown in [10] (p. 85), the spread with respect to time is $s^2\sigma_t^2$, where $\sigma_t^2 = \int t^2|\psi(t)|^2dt$ is the variance of ψ. Since ψ is analytic, $\Psi(\omega)$ is zero at negative frequencies, so the center frequency of Ψ is

$$\eta = \int_0^{+\infty}\omega|\Psi(\omega)|^2d\omega$$

The Fourier transform of $\psi_{u,s}$ is a dilation of Ψ by $1/s$:

$$\Psi_{u,s}(\omega) = \sqrt{s}\Psi(s\omega)e^{-i2\pi\omega u}$$

Its center frequency is therefore η/s. The energy spread of $\Psi_{u,s}$ around η/s is σ_ω^2/s^2 ([10] *ibid.*).

Thus the energy spread of a wavelet time-frequency atom $\psi_{u,s}$ corresponds to a Heisenberg box centered at $(u, \eta/s)$, of size $s\sigma_t$ along time and σ_ω/s along frequency. The area of the rectangle remains equal to $\sigma_t\sigma_\omega$ at all scales, but the resolution in time and frequency depends on s. As s increases (resp. decreases) frequency resolution increases (resp. decreases) while time resolution decreases (resp. increases).

An analytic wavelet transform defines a local time-frequency energy density $E_{\mathrm{W}[x]}$, which measures the energy of x in the Heisenberg box of each wavelet $\psi_{u,s}$ centered at $(t, \xi) = (u, \eta/s)$:

$$E_{\mathrm{W}[x]}(t,\xi) = |\mathrm{W}[x](u,s)|^2 = \left|\mathrm{W}[x]\left(u, \frac{\eta}{\xi}\right)\right|^2 \tag{10}$$

This energy density is called a *scalogram*.

3.4 Instantaneous Frequency

A cosine modulation

$$x(t) = a(t)\cos(\omega_0 t + \phi_0) = a(t)\cos\phi(t), \quad \text{for } a(t) \geq 0$$

has a frequency ω_0 that is the derivative of the phase $\phi(t) = \omega_0 t + \phi_0$. The *instantaneous frequency* is defined as a positive derivative of the phase:

$$\omega(t) = \phi'(t) \geq 0. \tag{11}$$

By adapting the sign of $\phi(t)$, the derivative can be taken to be positive. Since there are many possible choices of $a(t)$ and $\phi(t)$, it is clear that $\omega(t)$ is not uniquely defined relative to x.

Recall that a function $x_a \in L^2(\mathbb{R})$ is said to be *analytic* if its Fourier transform is zero for negative frequencies:

$$X_a(\omega) = 0 \text{ if } \omega < 0.$$

An analytic function is necessarily complex but is entirely characterized by its real part, $x = \mathrm{Re}[x_a]$, which has the Fourier transform

$$X(\omega) = \frac{X_a(\omega) + X_a^*(-\omega)}{2}$$

and this relation can be inverted to yield

$$X_a(\omega) = \begin{cases} 2X(\omega), & \text{if } \omega \geq 0 \\ 0, & \text{if } \omega < 0 \end{cases} \tag{12}$$

The analytic part $x_a(t)$ of a signal $x(t)$ is the inverse Fourier transform of $X_a(\omega)$ defined by (9). Writing the analytic part of x as the modulus times the complex exponential,

$$x_a(t) = a(t)\exp[i\phi(t)] = a(t)[\cos\phi(t) + i\sin\phi(t)]$$

it is clear that

$$x(t) = \mathrm{Re}[x_a] = a(t)\cos\phi(t)$$

The factor $a(t)$ is the *analytic* amplitude of $x(t)$, while $\phi'(t)$ is the instantaneous frequency. Both $a(t)$ and $\phi'(t)$ are uniquely defined.

Example 3.1 [3] If $x(t) = a(t)\cos(\omega_0 t + \phi_0)$, then

$$X(\omega) = \frac{1}{2}\left(\hat{a}(\omega - \omega_0)e^{i\phi_0} + \hat{a}(\omega + \omega_0)e^{-i\phi_0}\right) \tag{13}$$

where $\hat{a}(\omega)$ denotes the Fourier transform of $a(t)$. If the variations of $a(t)$ are slow compared to the period $T = 2\pi/\omega$, which is achieved by requiring that the support of $\hat{a}$ be included in $[-\omega_0, \omega_0]$, then the second term on the right of (13) is zero for $\omega > 0$ and

$$X_a(t) = \hat{a}(\omega - \omega_0)e^{i\phi_0}$$

by (9). So $x_a(t) = a(t)\exp[i(\omega_0 t + \phi_0)]$.

Example 3.2 If a signal x is the sum of two sinusoidal waves, $a\cos(\omega_1 t) + a\cos(\omega_2 t)$, then

$$\begin{aligned} x_a(t) &= ae^{i\omega_1 t} + ae^{i\omega_2 t} \\ &= a\cos[(\omega_1 - \omega_2)t/2]\exp[i(\omega_1 + \omega_2)t/2] \end{aligned}$$

The instantaneous frequency is $\phi'(t) = (\omega_1 + \omega_2)/2$ and the amplitude is

$$a(t) = a\,|\cos[(\omega_1 - \omega_2)t/2]|$$

This result is not satisfying because it does not reveal that the signal includes two sinusoidal waves of the same amplitude. It measures an average frequency value. The next sections explain how to measure the instantaneous frequencies of several spectral components by separating them with a windowed Fourier transform or a wavelet transform.

[3] Mallat [10] (p.92)

3.5 Fourier Ridges

The spectrogram $E_{\mathrm{S}[x]}(u,\xi)$ measures the energy of x in a time-frequency neighborhood of (u,ξ). The ridge algorithm computes the instantaneous frequencies from the local maxima of $E_{\mathrm{S}[x]}(u,\xi)$. This approach was introduced by Delprat, Escudié, Guillemain, Kronland-Martinet, Tchamitchian and Torrésani [11] to analyze musical sounds. Since then, it has found applications for a wide range of signals that have time varying frequency tones.

When the signal contains several spectral lines *whose frequencies are sufficiently apart*, the windowed Fourier transform separates each of these components and the ridges detect the evolution in time of each spectral component.

The windowed Fourier transform is computed with a symmetric window $g(t) = g(-t)$ with support $t \in [-1/2, 1/2]$ and unit norm, $\|g\| = 1$. For a fixed scale s, $g_s(t) = s^{-1/2} g(t/s)$ has a support of size s and unit norm. The corresponding windowed Fourier atoms are $g_{s,u,\xi}(t) = g_s(t-u)e^{i\xi t}$ and the windowed Fourier transform is defined by

$$\mathrm{S}[x](u,\xi) = \int_{-\infty}^{\infty} x(t) g_s(t-u) \mathrm{e}^{-i\xi t} dt$$

If $x(t) = a(t) cos\phi(t)$, then the transform $\mathrm{S}[x](u,\xi)$ has the following relation to the instantaneous frequency of x:

$$\mathrm{S}[x](u,\xi) = \frac{\sqrt{s}}{2} a(u) \exp(i[\phi(u) - \xi u]) \left(\hat{g}(s[\xi - \phi'(u)]) + \epsilon\right) \tag{14}$$

for all $\xi \geq 0$. For bounds on ϵ, see Mallat [10]. Delprat *et al.* [11] prove a similar result for Gaussian g.

If we can neglect the corrective term (ϵ) then (14) enables us to measure $a(u)$ and $\phi'(u)$ using $\mathrm{S}[x](u,\xi)$. Briefly, the corrective term is negligible when two conditions are satisfied. First, let the bandwidth of $\hat{g}$ be $\Delta\omega$ such that $|\hat{g}(\omega)| \ll 1$ when $|\omega| \geq \Delta\omega$. Then we require that

$$\phi'(u) \geq \frac{\Delta\omega}{s}$$

The second condition permitting us to neglect ϵ is that $a(t)$ and $\phi'(t)$ must have small relative variations over the support of the scaled window g_s. If these two conditions are met, then (14) shows that for each u, the spectrogram $|\mathrm{S}[x](u,\xi)|^2$ is maximized at $\xi(u) = \phi'(u)$. The corresponding time-frequency pairs $(u, \xi(u))$ are called *ridge points*. Such points indicate that the predominant spectral components of the signal occur at frequencies $\phi'(u) = \xi(u)$ with analytic amplitudes, computed from (14),

$$a(u) = \frac{2\,|\mathrm{S}[x](u,\xi(u)|}{\sqrt{s}|\hat{g}(0)|}$$

When the signal contains several spectral lines *whose frequencies are sufficiently apart*, the windowed Fourier transform separates each of these components and the ridges detect the evolution in time of each spectral component. Suppose, for instance, that x has the form

$$x(t) = a_1(t)\cos\phi_1(t) + a_2(t)\cos\phi_2(t)$$

where $a_k(t)$ and $\phi_k'(t)$ have small variations over intervals of size s and $s\phi_k'(t) \geq \Delta\omega$. Since the Fourier transform is a linear operation, equation (14) implies that

$$\mathrm{S}[x](u,\xi) \approx \frac{\sqrt{s}}{2} a_1(u) \exp(i[\phi_1(u) - \xi u]) \hat{g}(s[\xi - \phi_1'(u)]) \tag{15}$$

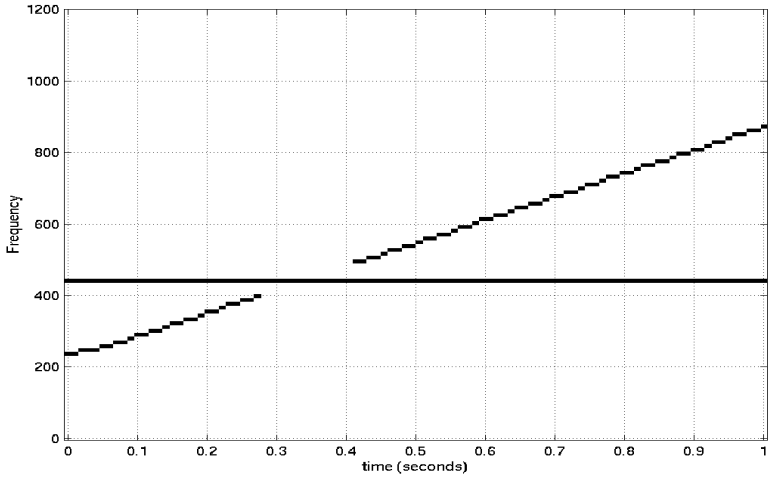

Figure 5: The local maximum moduli of the spectrogram are the ridge points that locate instantaneous frequencies.

$$+\frac{\sqrt{s}}{2}a_2(u)\exp(i[\phi_2(u)-\xi u])\hat{g}(s[\xi-\phi_2'(u)])$$

The two spectral components can be discriminated if for all u, $\hat{g}(s|\phi_1'(u)-\phi_2'(u)|) \ll 1$ which means that the frequency difference is larger than the bandwidth of $\hat{g}(s\omega)$:

$$|\phi_1'(u)-\phi_2'(u)| \geq \frac{\Delta\omega}{s} \tag{16}$$

In this case, when $\xi = \phi_1'(u)$, the second term of (15) can be neglected and the first term generates a ridge point from which we may recover $\phi_1'(u)$ and $a_1(u)$. Similarly for the case $\xi = \phi_2'(u)$. The ridge points are distributed along two time-frequency lines $\xi(u) = \phi_1'(u)$ and $\xi(u) = \phi_2'(u)$. This result is valid for any number of time-varying spectral components, as long as the distance between any two instantaneous frequencies satisfies (16). For those values of u at which the spectral lines are too close, they interfere and destroy the ridge pattern.

Figure 5 displays the ridges computed from the local maxima of the spectrogram of

$$x(t) = 0.5\cos(2\pi 440t) + 0.4\cos(2\pi\alpha 440t) \tag{17}$$

where $\alpha = 0.5 + 1.5t$. That is, the first summand is a pure 440 Hz tone with a constant amplitude of 0.5, while the second term has amplitude 0.4 and a frequency that increases linearly from 220 Hz to 880 Hz over the interval $t \in [0, 1]$. The ridges fail to accurately describe the frequency content of the signal when t ranges from about 0.3 to 0.4 seconds. Here the instantaneous frequencies are too close and the frequency resolution is not sufficient to distinguish them.

Wavelet Ridges. Windowed Fourier atoms have a fixed scale and thus cannot follow the instantaneous frequencies of rapidly varying events such as hyperbolic chirps. In contrast analytic

wavelet transform modifies the scale of its time-frequency atoms. The ridge algorithm of Delprat *et al* (*op. cit.*) can be extended to analytic wavelet transforms to accurately measure frequency tones that are rapidly changing at high frequencies.

3.6 Matching Pursuit

A *matching pursuit* [12] is an iterative algorithm that decomposes the signal over *dictionary* vectors. A dictionary is a family of vectors $\mathcal{D} = \{g_\gamma\}_{\gamma \in \Gamma}$ included in a Hilbert space $\mathcal{H}$, with a unit norm $\|g_\gamma\| = 1$. Such a family can be constructed by scaling, translating and modulating a single window function $g(t) \in L^2(\mathbb{R})$. We suppose that $g(t)$ is real, continuously differentiable and $O(\frac{1}{t^2+1})$. We further impose that $\|g\| = 1$, that the integral of $g(t)$ is non-zero, and that $g(0) \neq 0$. For any scale s, translation u, and frequency modulation ξ, we denote $\gamma = (u, s, \xi)$ and define

$$g_\gamma(t) = \frac{1}{\sqrt{s}} g\left(\frac{t-u}{s}\right) e^{i2\pi\xi t} \tag{18}$$

The index γ is an element of the set $\mathbf{\Gamma} = \mathrm{Re} \times \mathrm{Re}^+ \times \mathrm{Re}$. The factor $1/\sqrt{s}$ normalizes the norm of g_γ to 1. If $g(t)$ is even, which is generally the case, $g_\gamma(t)$ is centered at the abscissa u. Its energy is mostly concentrated in a neighborhood of u, whose size is proportional to s. Let $\hat{g}(\omega)$ be the Fourier transform of $g(t)$. Equation (18) yields

$$\hat{g}_\gamma(\omega) = \sqrt{s}\hat{g}(s(\omega - \xi))e^{-i2\pi(\omega-\xi)u}$$

The matching pursuit algorithm iteratively decomposes a signal over dictionary vectors as follows. Let $R^0x(t) = x(t)$, and suppose that we have computed the n^{th} order *residue*, $R^n x$, for $n \geq 0$. We then choose an element, g_{γ_n} which closely "matches" the residue in the following sense:

$$|C(R^n x, g_{\gamma_n})| = \sup_{\gamma \in \Gamma} |C(R^n x, g_\gamma)|$$

where $C(x, g_\gamma)$ is a correlation function which measures the similarity between x and g_γ. An example is the usual inner product, $\langle x, g_\gamma \rangle$.

Next, decompose the residue as

$$R^n x(t) = C(R^n x, g_{\gamma_n}) g_{\gamma_n}(t) + C(R^{n+1} x(t)$$

which defines the residue for step $n+1$, and fully specifies the algorithm recursion.

With the usual inner product as the correlation function, it can be shown ([10], p.422) that the magnitude of the residue, $\|R^n x\|$ converges to 0 exponentially as n increases. This yields the following atomic signal decomposition:

$$x(t) = \sum_{n=0}^{\infty} C(R^n x, g_{\gamma_n}) g_{\gamma_n}(t) \tag{19}$$

4 Energy Distributions

Wavelet and windowed Fourier transforms are computed by correlating the signal with families of time-frequency atoms. The time and frequency resolution of these transforms is thus limited by the time-frequency resolution of the corresponding atoms. Ideally, one would like to define a density of energy in a time-frequency plane with no loss of resolution. This section presents a different class of time-frequency representation (TFR) which is not restricted by the uncertainty principle.

The *Wigner-Ville time-frequency representation* is computed by correlating x with a time and frequency translation of itself. (Below we refer to the Wigner-Ville TFR simply as the "Wigner-Ville" and, sometimes, as the "Wigner transform.") Though it yields some remarkable properties, the quadratic from of this representation can also limit its application because of the inevitable cross terms that appear in quadratic forms. In such cases, these so-called "interference terms" can be attenuated by a time-frequency averaging, but this procedure results in a loss of resolution. Below we see that the spectrogram, the scalogram and all squared time-frequency decompositions can be written as a time-frequency averaging of the Wigner transform.

Although the TFR's of this section do not yield positive distributions in all cases, they allow extremely good insight into signal properties within certain applications.

4.1 Ambiguity Function

This section describes TFR's that are computed by correlating a signal, $x(t)$, with time and frequency shifted versions of itself. We start by looking at time and frequency shifts separately.

Time-Shifted Signals. We define the *temporal autocorrelation function*, $r_x(\tau)$, by correlating $x(t)$ with a time translation of itself:

$$r_x(\tau) = \langle x_\tau, x \rangle = \int_{-\infty}^{\infty} x(t+\tau) x^*(t)\, dt \tag{20}$$

A simple change of variable shows that $r_x(\tau) = \langle x, x_{-\tau} \rangle$.

The distance $d(x, x_\tau)$ between an energy signal $x(t)$ and its time-shifted version $x_\tau(t) = x(t+\tau)$ is related to the autocorrelation function according to the following equation:

$$d(x, x_\tau)^2 = 2\|x\|^2 - 2\,\mathrm{Re}[r_x(\tau)]$$

Let $Y(\omega)$ denote the Fourier transform of $y(t)$ and recall the following correspondences:

	$y(t)$	$Y(\omega)$
translation	$x(t-t_0)$	$e^{-i2\pi\omega t_0} X(\omega)$
modulation	$e^{i2\pi\omega_0 t} x(t)$	$X(\omega-\omega_0)$
conjugation	$x^*(t)$	$X^*(-\omega)$

Also, recall Parseval's relation from section A.4: $\langle X, Y\rangle = \langle x, y\rangle$. Now, if we put $y(t) = x(t+\tau)$ in equation (20), it is easy to see that $r_x(\tau) = \langle y, x\rangle = \langle Y, X\rangle$. More explicitly, we have

$$\begin{aligned} r_x(\tau) &= \int_{-\infty}^{\infty} Y(\omega)X^*(\omega)\,d\omega \\ &= \int_{-\infty}^{\infty} e^{i2\pi\omega\tau}X(\omega)X^*(\omega)\,d\omega \\ &= \int_{-\infty}^{\infty} |X(\omega)|^2\, e^{i2\pi\omega\tau}\,d\omega \end{aligned} \tag{21}$$

where $|X(\omega)|^2$ is the *spectral energy density.* Thus, inverting (21), we see that the spectral energy density is the Fourier transform of the temporal autocorrelation function.

Frequency-Shifted Signals. Frequency shifted versions of a signal $x(t)$ are often produced due to the Doppler effect. If we wish to estimate such a frequency shift in order to determine the velocity of a moving object, we consider the distance between a signal $x(t)$ and its frequency-shifted, or *modulated*, version $x_\xi(t) = x(t)e^{i2\pi\xi t}$. The distance is given by

$$d(x, x_\xi)^2 = 2\|x\|^2 - 2\,\mathrm{Re}\langle x_\xi, x\rangle$$

The foregoing inner product will be denoted $\rho_x(\xi)$. Thus,

$$\begin{aligned} \rho_x(\xi) = \langle x_\xi, x\rangle &= \langle x, x_{-\xi}\rangle \\ &= \int_{-\infty}^{\infty} e^{i2\pi\xi t}x(t)x^*(t)\,dt \\ &= \int_{-\infty}^{\infty} |x(t)|^2\, e^{i2\pi\xi t}\,dt \end{aligned}$$

where $|x(t)|^2$ is the *temporal energy density.* It is also called the *instantaneous power* of the signal.

It is possible to view $\rho_x(\xi)$ as the spectral analog of the temporal autocorrelation function, $r_x(\tau)$, when we write the former in as a *spectral autocorrelation function.* That is, using the Fourier relations reviewed above and the Parseval identity,

$$\rho_x(\xi) = \int_{-\infty}^{\infty} X(\omega - \xi)X^*(\omega)\,d\omega$$

In other words, $\rho_x(\xi)$ is the autocorrelation function of the spectrum $X(\omega)$.

Time and Frequency-Shifted Signals. We now consider shifting a signal in both the time and frequency domains simultaneously. For this, a transform known as the *ambiguity function* plays a central role as the *time-frequency autocorrelation function.* It is defined as follows:

$$\mathrm{A}_x(\xi, \tau) = \int_{-\infty}^{\infty} x\left(t + \tfrac{\tau}{2}\right) x^*\left(t - \tfrac{\tau}{2}\right) e^{i2\pi\xi t}\,dt$$

Once again, Parseval provides for the definition as a frequency integration

$$\mathrm{A}_x(\xi, \tau) = \int_{-\infty}^{\infty} X\left(\nu - \tfrac{\xi}{2}\right) X^*\left(\nu + \tfrac{\xi}{2}\right) e^{i2\pi\nu\tau}\,d\nu$$

Analogous to the cross correlation function is the *cross ambiguity function* defined by

$$\begin{aligned} \mathrm{A}_{xy}(\xi,\tau) &= \int_{-\infty}^{\infty} x\left(t+\tfrac{\tau}{2}\right) y^*\left(t-\tfrac{\tau}{2}\right) e^{i2\pi\xi t}\, dt \qquad (22) \\ &= \int_{-\infty}^{\infty} X\left(\nu-\tfrac{\xi}{2}\right) Y^*\left(\nu+\tfrac{\xi}{2}\right) e^{i2\pi\xi t}\, dt \qquad (23) \end{aligned}$$

At this point it is helpful to summarize what we have shown thus far. The functions $r_x(\tau)$ and $\rho_x(\xi)$ are the autocorrelation functions associated with shifts in time and frequency, respectively. The *spectral* energy density is the Fourier transform of the *temporal* autocorrelation function:

$$|X(\nu)|^2 = \int_{-\infty}^{\infty} r_x(\tau) e^{-i2\pi\nu\tau}\, d\tau$$

The *temporal* energy density is the Fourier transform of the *spectral* autocorrelation function:

$$|x(t)|^2 = \int_{-\infty}^{\infty} \rho_x(\xi) e^{-i2\pi\xi t}\, d\xi$$

The spectral and temporal energy densities can be viewed as "marginal" energy densities in their respective variables. The ambiguity function generalizes the "marginal" autocorrelation functions, $r_x(t)$ and $\rho_x(\xi)$, to simultaneously account for both time and frequency shifts. Thus, it is the "joint" autocorrelation function. The natural next step, then, is to consider the Fourier transform of the ambiguity function. Perhaps this defines a useful "joint" energy density, with the correct marginal densities, $|X(\nu)|^2$ and $|x(t)|^2$. We define such a transform in the next section, and relate it to the foregoing interpretation as a "joint energy density" in section 4.3.

4.2 Wigner Transform

The ambiguity function represents the signal energy in the *frequency-time* plane. We now transform this representation into the *time-frequency* plane.

The quadratic form

$$\mathrm{W}_x(t,\nu) = \int_{-\infty}^{\infty} x\left(t+\tfrac{\tau}{2}\right) x^*\left(t-\tfrac{\tau}{2}\right) e^{-i2\pi\tau\nu}\, d\tau \qquad (24)$$

is known as the *Wigner-Ville distribution*, or *Wigner transform*. As we show in the next section, it is the two-dimensional Fourier transform of the ambiguity function. Also, as the one-dimensional Fourier transform of $x^*\left(t-\frac{\tau}{2}\right) x\left(t+\frac{\tau}{2}\right)$, which has a Hermitian symmetry in τ, the Wigner-Ville transform is real valued.

Time and frequency play a symmetric role, and the transform can be written as a frequency integral by applying the Parseval formula:

$$\mathrm{W}_x(t,\nu) = \int_{-\infty}^{\infty} X\left(\nu-\tfrac{\xi}{2}\right) X^*\left(\nu+\tfrac{\xi}{2}\right) \mathrm{e}^{-i2\pi\xi t}\, dt \qquad (25)$$

The Wigner-Ville transform localizes the time-frequency structures of x. If the energy of x is well concentrated in time around t_0 and in frequency around ν_0 then W_x has its energy centered at (t_0,ν_0), with a spread equal to the time and frequency spread of x.

Instantaneous Frequency. Ville's original motivation for studying time-frequency decompositions was to compute the instantaneous frequency of a signal. Assume x has the form $x(t) = a(t)\cos\phi(t)$ and let x_a be the analytic part of x obtained by setting $X(\nu) = 0$ for $\nu < 0$. Equivalently, $x_a(t) = a(t)\exp[i\phi(t)]$ so that $x = \mathrm{Re}[x_a]$, and $\nu = \phi'(t)$ is the instantaneous frequency. The following proposition states that $\phi'(t)$ is the "average" frequency computed relative to the transform W_x.

Proposition 4.1 If $x_a(t) = a(t)\exp[i\phi(t)]$, then

$$\phi'(t) = \frac{\int \nu\, \mathrm{W}_{x_a}(t,\nu)\, d\nu}{\int \mathrm{W}_{x_a}(t,\nu)\, d\nu}$$

where integration is over all $\mathbb{R}$.

This proposition shows that, for a fixed $t = t_0$, the mass of $\mathrm{W}_{x_a}(t_0,\nu)$ is typically concentrated in the neighborhood of the instantaneous frequency $\nu_0 = \phi'(t)|_{t=t_0}$.

4.3 Wigner-Ambiguity Relations

To help further motivate the definition of the Wigner-Ville, we relate it to other entities, about which we might have better intuition. First, let us simplify notation by introducing the signals

$$x_{\frac{\tau}{2},\frac{\xi}{2}}(t) = x\left(t+\tfrac{\tau}{2}\right)e^{i2\pi\frac{\xi}{2}t}, \qquad x_{-\frac{\tau}{2},-\frac{\xi}{2}}(t) = x\left(t-\tfrac{\tau}{2}\right)e^{-i2\pi\frac{\xi}{2}t} \tag{26}$$

$$y_{t,\nu}(\tau) = x\left(t+\tfrac{\tau}{2}\right)e^{-i2\pi\nu\frac{\tau}{2}}, \qquad y_{t,\nu}(-\tau) = x\left(t-\tfrac{\tau}{2}\right)e^{i2\pi\nu\frac{\tau}{2}} \tag{27}$$

which are time and frequency shifted versions of $x(t)$. Note that, because of its role in defining the Wigner-Ville transform, the function $y_{t,\nu}$ is defined as a function of the delay variable τ.

The notations in (26) and (27) allow us to define the ambiguity function and the Wigner-Ville as inner products,

$$\begin{aligned} \mathrm{A}_x(\xi,\tau) &= \langle x_{\frac{\tau}{2},\frac{\xi}{2}}, x_{-\frac{\tau}{2},-\frac{\xi}{2}}\rangle & (28)\\ \mathrm{W}_x(t,\nu) &= \langle y_{t,\nu}, \tilde{y}_{t,\nu}\rangle & (29)\end{aligned}$$

where $\tilde{y}_{t,\nu}(\tau) = y_{t,\nu}(-\tau)$. This notation elucidates the role of the transforms in relation to the distance between shifted signals.

$$\begin{aligned} d(x_{-\frac{\tau}{2},-\frac{\xi}{2}}, x_{\frac{\tau}{2},\frac{\xi}{2}}) &= 2\|x\|^2 - 2\,\mathrm{Re}\langle x_{\frac{\tau}{2},\frac{\xi}{2}}, x_{-\frac{\tau}{2},-\frac{\xi}{2}}\rangle \\ &= 2\|x\|^2 - 2\,\mathrm{Re}[\mathrm{A}_x(\xi,\tau)] \end{aligned}$$

$$\begin{aligned} d(\tilde{y}_{t,\nu}, y_{t,\nu}) &= 2\|x\|^2 - 2\,\mathrm{Re}\langle y_{t,\nu}, \tilde{y}_{t,\nu}\rangle \\ &= 2\|x\|^2 - 2\,\mathrm{Re}[\mathrm{W}_x(t,\nu)] \end{aligned}$$

Next, recall the temporal and spectral autocorrelation functions of section 4.1

$$r_x(\tau) = \langle x_\tau, x\rangle = \int_{-\infty}^{\infty} x(t+\tau)x^*(t)\, dt$$

$$\rho_x(\xi) = \langle x_\xi, x\rangle = \int_{-\infty}^{\infty} X(\omega-\xi)X^*(\omega)\, d\omega$$

and restrict the ambiguity function to the "delay axis," $\{(\xi,\tau) : \xi = 0\}$ – that is, fix the Doppler variable at $\xi \equiv 0$. On the delay axis the ambiguity function is identical to the temporal autocorrelation function:

$$\mathrm{A}_x(0,\tau) = r_x(\tau)$$

From this we derive the spectral energy density by means of the following Fourier transform:

$$\begin{aligned} |X(\nu)|^2 &= \int_{-\infty}^{\infty} r_x(\tau) e^{-i2\pi\nu\tau}\, d\tau \\ &= \int_{-\infty}^{\infty} \mathrm{A}_x(0,\tau) e^{-i2\pi\nu\tau}\, d\tau \end{aligned}$$

On the other hand, by projecting $\mathrm{A}_x(\xi,\tau)$ onto the Doppler axis, where $(\xi,\tau) \equiv (\xi,0)$, we arrive at the autocorrelation function of the spectrum:

$$\mathrm{A}_x(\xi,0) = \rho_x(\xi)$$

The temporal energy density $|x(t)|^2$ is the Fourier transform of $\rho_x(\nu)$:

$$\begin{aligned} |x(t)|^2 &= \int_{-\infty}^{\infty} \rho_x(\nu) e^{-i2\pi\xi t}\, d\xi \\ &= \int_{-\infty}^{\infty} \mathrm{A}_x(\nu,0) e^{-i2\pi\xi t}\, d\xi \end{aligned}$$

On the entire frequency-time plane, the Fourier transform with respect to ξ yields the product:

$$\begin{aligned} \phi_x(t,\tau) &= \int_{-\infty}^{\infty} \mathrm{A}_x(\xi,\tau) e^{-i2\pi\xi t}\, d\xi \\ &= x\left(t+\tfrac{\tau}{2}\right) x^*\left(t-\tfrac{\tau}{2}\right) \end{aligned} \tag{30}$$

The Fourier transform with respect to τ yields

$$\begin{aligned} \Phi_x(\xi,\nu) &= \int_{-\infty}^{\infty} \mathrm{A}_x(\xi,\tau) e^{-i2\pi\nu\tau}\, d\tau \\ &= X\left(\nu-\tfrac{\xi}{2}\right) X^*\left(\nu+\tfrac{\xi}{2}\right) \end{aligned} \tag{31}$$

By equations (30) and (31), and the definition of the Wigner-Ville transform (24), we have the following two-dimensional Fourier transform:

$$\mathrm{W}_x(t,\nu) = \iint\limits_{\mathbb{R}^2} \mathrm{A}_x(\xi,\tau) e^{-i2\pi(\xi t+\nu\tau)}\, d\xi\, d\tau \tag{32}$$

The foregoing also makes clear the connections between the Wigner-Ville transform and the energy densities $|x(t)|^2$ and $|X(\nu)|^2$. Indeed, we have the Fourier relations

$$\begin{aligned} \phi_x(t,\tau) &= \int_{-\infty}^{\infty} \mathrm{W}_x(t,\nu) e^{i2\pi\nu\tau}\, d\nu \\ \Phi_x(\xi,\nu) &= \int_{-\infty}^{\infty} \mathrm{W}_x(t,\nu) e^{i2\pi\xi t}\, dt \end{aligned} \tag{33}$$

Restricting $\phi(t,\tau)$ to the set $\{(t,\tau) : \tau = 0\}$ in equation (30) yields

$$|x(t)|^2 = \phi_x(t,0) = \int_{-\infty}^{\infty} \mathrm{W}_x(t,\nu)\, d\nu$$

Similarly, equations (30) and (33) imply

$$|X(\nu)|^2 = \Phi_x(0,\nu) = \int_{-\infty}^{\infty} \mathrm{W}_x(t,\nu)\, dt$$

Thus we have proved that the "marginal" temporal energy density is recovered by integrating the Wigner transform with respect to the frequency variable. Similarly, the "marginal" spectral energy density is recovered by an integration with respect to time. Therefore, it is intuitively appealing to call W_x a "joint time-frequency energy density." However, the Wigner-Ville transform misses one fundamental property of a density function – positivity. In general, the Wigner-Ville is an oscillating function that takes on negative values. In fact, one can prove that translated and frequency modulated Gaussians are the only functions whose Wigner-Ville transforms remain positive.

The negative values, popularly called "interferences," are created by the quadratic properties of the Wigner-Ville transform. These interferences can be attenuated or removed by averaging the transform with appropriate kernels which yield positive time-frequency densities. However, this reduces the time-frequency resolution. The spectrograms of Fourier analyses as well as the scalograms of wavelet analyses are examples of positive quadratic densities obtained by smoothing the Wigner transform. In fact, for any family $\{\phi_\gamma\}$ of time-frequency atoms, and the associated transform $\mathrm{T}[x]$, the energy density $E_{T[x]}(t,\nu)$, is an averaging of the Wigner transform. We will consider these ideas more carefully in the following sections.

We conclude this section by summarizing the relationships between the Wigner-Ville, ambiguity, and (deterministic)[4] autocorrelation functions using the following diagram:

$$\begin{array}{ccc} \phi_x(t,\tau) & \xleftarrow{\mathcal{F}} & A_x(\xi,\tau) \\ {\scriptstyle\mathcal{F}}\downarrow & & \downarrow{\scriptstyle\mathcal{F}} \\ W_x(t,\nu) & \xleftarrow[\mathcal{F}]{} & \Phi_x(\xi,\nu) \end{array} \tag{34}$$

Here $\mathcal{F}$ indicates that the Fourier transform operates in the direction of the arrow.

The Diagram is helpful for remembering the Fourier relationships because it agrees with the standard geometrical framework in which a change in the first (resp. second) coordinate corresponds to movement along the horizontal (resp. vertical) axis. However, this interpretation cannot be taken literally since movement along each axis really corresponds to an entire change of coordinates. The next diagram shows the coordinate systems in which the associated objects of the previous diagram represent the signal energy:

$$\begin{array}{ccc} \text{time-time} & \xleftarrow{\mathcal{F}} & \text{frequency-time} \\ {\scriptstyle\mathcal{F}}\downarrow & & \downarrow{\scriptstyle\mathcal{F}} \\ \text{time-frequency} & \xleftarrow[\mathcal{F}]{} & \text{frequency-frequency} \end{array}$$

[4]If $x(t)$ was assumed to be a random process, $\mathrm{E}\{\phi_x(t,\tau)\}$ would be the autocorrelation function of the process.

For further insight into the different perspectives that each of these parameterizations provide, we refer the reader to Flandrin's excellent treatment in [13], pages 187–194.

4.4 Interference Structure

Because the Wigner-Ville transform is a sesquilinear form of the signal, it cannot submit to the principle of linear superposition. As in the quadratic equation,

$$(a+b)^2 = a^2 + b^2 + ab + ba$$

it is easy to verify that

$$\mathrm{W}_{x+y}(t,\nu) = \mathrm{W}_x(t,\nu) + \mathrm{W}_y(t,\nu) + \mathrm{W}_{xy}(t,\nu) + \mathrm{W}_{yx}(t,\nu) \tag{35}$$

where W_{xy} is the cross Wigner-Ville transform of two signals, defined by

$$\begin{aligned} \mathrm{W}_{xy}(t,\nu) &= \int_{-\infty}^{\infty} x\left(t+\tfrac{\tau}{2}\right) y^*\left(t-\tfrac{\tau}{2}\right) e^{-i2\pi\nu\tau}\, d\tau \\ &= \int_{-\infty}^{\infty} X\left(\nu-\tfrac{\xi}{2}\right) Y^*\left(\nu+\tfrac{\xi}{2}\right) e^{-i2\pi\xi t}\, d\xi \end{aligned} \tag{36}$$

It can also be shown that (36) is the two-dimensional Fourier transform of the cross ambiguity function defined above in equation (22).

We define the *interference term* of equation (35) by

$$\begin{aligned} I_{xy}(t,\nu) &= \mathrm{W}_{xy}(t,\nu) + \mathrm{W}_{yx}(t,\nu) \\ &= 2\,\mathrm{Re}\left[\mathrm{W}_{xy}(t,\nu)\right] \end{aligned}$$

It is a real valued function that creates non-zero values at unexpected locations of the time-frequency plane. In general, for any linear combination of signal components,

$$x(t) = \sum_{n=1}^{N} a_n x_n(t)$$

the Wigner-Ville transform is

$$\mathrm{W}_x = \sum_{n=1}^{N} |a_n|^2\, \mathrm{W}_{x_n}(t,\nu) + 2\sum_{n=1}^{N-1}\sum_{k=n+1}^{N} \mathrm{Re}\left[a_n a_k^*\, \mathrm{W}_{x_n x_k}(t,\nu)\right] \tag{37}$$

Hence, for a signal with N components, the Wigner-Ville transform contains $N(N-1)/2$ additional components. They result from the interaction of different components of the signal, and are called "interference terms" for two reasons. First, the mechanism of their creation is analogous to the usual interference, which can be observed for physical waves. A second reason for this terminology lies in the disturbing effect that these terms can have on the time-frequency diagram of the signal energy. As they amount to a combinatorial proliferation of additional, "specious" signal components, they can inhibit our ability to discern "true" signal components in the diagram.

The presence of cross terms in a Wigner-Ville transform can be regarded as a natural consequence of its bilinear structure. On the other hand, this very structure also leads to most of the good

properties of the transform (such as localization). No matter whether one views the cross terms as helpful or hindering, it is important to understand fully the mechanism of their creation. This is indispensable for drawing the correct interpretation from the representation of an unknown signal, and for reducing the importance of these terms if desired. In our present work, we study the cross terms in order to understand how this measure of signal interference relates to measures of "musical interference," e.g. dissonance.

The canonical example used to describe the structure of the cross terms begins with a well localized time-frequency atom $x(t)$ centered at $t = 0$. From this we construct two atoms which are time and frequency shifted versions of $x(t)$,

$$x_1(t) = a_1 x(t - t_1) e^{i2\pi(\nu_1 t + \phi_1)}, \qquad a_1 \geq 0$$
$$x_2(t) = a_2 x(t - t_2) e^{i2\pi(\nu_2 t + \phi_2)}, \qquad a_2 \geq 0$$

Now consider the Wigner-Ville transform of the composite signal $x_1(t) + x_2(t)$,

$$\mathrm{W}_{x_1+x_2}(t,\nu) = \mathrm{W}_{x_1}(t,\nu) + \mathrm{W}_{x_2}(t,\nu) + I_{x_1 x_2}(t,\nu)$$

The *covariance property* of the Wigner-Ville transform ensures that the shifted atoms, taken individually, have Wigner representations given by

$$\mathrm{W}_{x_1}(t,\nu) = a_1^2 \, \mathrm{W}_x(t - t_1, \nu - \nu_1)$$
$$\mathrm{W}_{x_2}(t,\nu) = a_2^2 \, \mathrm{W}_x(t - t_2, \nu - \nu_2)$$

Since the energy of W_x is centered at $(0, 0)$, the energy of W_{x_1} and W_{x_2} is concentrated in the neighborhoods (t_1, ν_1) and (t_2, ν_2), respectively. A direct calculation verifies that the interference term is

$$I_{x_1,x_2}(t,\nu) = 2a_1 a_2 \, \mathrm{W}_x(t - t_m, \nu - \nu_m) \cos\left\{2\pi\left[(t - t_m)\Delta\nu - (\nu - \nu_m)\Delta t + \Delta\phi\right]\right\}$$

where

$$t_m = \frac{t_1 + t_2}{2}, \quad \nu_m = \frac{\nu_1 + \nu_2}{2}$$
$$\Delta t = t_1 - t_2, \quad \Delta\nu = \nu_1 - \nu_2$$
$$\Delta\phi = \phi_1 - \phi_2 + t_m \Delta\nu$$

This is an oscillatory waveform concentrated in a neighborhood of the point in the time-frequency plane that is the geometric midpoint between the individual components. The frequency of the oscillations is proportional to the Euclidean distance $\sqrt{\Delta\nu^2 + \Delta t^2}$ that separates the points (t_1, ν_1) and (t_2, ν_2), where the individual atoms are concentrated. The direction of these oscillations is perpendicular to the line that joins these two center points.

Physical Interpretation. It is possible to attach physical meaning to the interference structure of the Wigner-Ville transform. For the most basic case, in which the signal is a simple superposition of pure frequencies, the cross term can be regarded as a signature of the *beat frequency* resulting from the interaction between the individual frequencies. For example, suppose we start with a pure sinusoidal $x(t) = e^{i2\pi\nu_m t}$ at the (mid-point) frequency ν_m. Then consider the two frequency shifted versions of $x(t)$,

$$x_1(t) = e^{i2\pi(\nu_m - \frac{\Delta\nu}{2})t}, \qquad x_2(t) = e^{i2\pi(\nu_m + \frac{\Delta\nu}{2})t}$$

The Wigner-Ville transform of the signal $x_1(t) + x_2(t)$ is given by

$$\begin{aligned} \mathrm{W}_{x_1+x_2}(t,\nu) &= \mathrm{W}_{x_1}(t,\nu) + \mathrm{W}_{x_2}(t,\nu) + I_{x_1x_2}(t,\nu) \\ &= \delta(\nu - (\nu_m - \tfrac{\Delta\nu}{2})) + \delta(\nu - (\nu_m + \tfrac{\Delta\nu}{2})) + \delta(\nu - \nu_m)\, 2\cos(2\pi\Delta\nu t) \end{aligned} \tag{38}$$

Now let us relate this expression to the physical phenomenon of beats, which are perceived most easily when the distance between signal components is small. To do so, we write the signal as follows:

$$\begin{aligned} x_1(t) + x_2(t) &= e^{i2\pi(\nu_m - \frac{\Delta\nu}{2})t} + e^{i2\pi(\nu_m + \frac{\Delta\nu}{2})t} \\ &= (e^{-i2\pi\frac{\Delta\nu}{2}t} + e^{i2\pi\frac{\Delta\nu}{2}t})\, e^{i2\pi\nu_m t} \\ &= 2\,\cos(2\pi\tfrac{\Delta\nu}{2}t)\, e^{i2\pi\nu_m t} \end{aligned} \tag{39}$$

When the components $x_1(t)$ and $x_2(t)$ are close together in frequency – that is, when $\Delta\nu$ is small – the cosine term is slowly varying as compared to the exponential term, and the resulting signal can be viewed as a simple tone of frequency ν_m with a modulated amplitude envelope, with modulation frequency $\Delta\nu$. The term "beats," or "beating," refers to such amplitude modulations.

Comparing (39) with (38), it is clearly the interference term of the Wigner-Ville transform which specifies the existence and nature of beats in the composite signal. To make for easier comparison, we can also write this signal as

$$x_1(t) + x_2(t) = \tfrac{1}{2}\, e^{i2\pi(\nu_m - \frac{\Delta\nu}{2})t} + \tfrac{1}{2}\, e^{i2\pi(\nu_m + \frac{\Delta\nu}{2})t} + \cos(2\pi\tfrac{\Delta\nu}{2}t)\, e^{i2\pi\nu_m t}$$

Interferences in the Ambiguity Plane. The ambiguity function is also a bilinear form, has its own interference structure, and represents both signal and interference components in the frequency-time, or "Doppler-delay," plane. A superposition of two signal components, $x_1(t)+x_2(t)$, yields the following frequency-time representation:

$$\mathrm{A}_{x_1+x_2}(\xi,\tau) = \mathrm{A}_{x_1}(\xi,\tau) + \mathrm{A}_{x_2}(\xi,\tau) + \mathrm{A}_{x_1x_2}(\xi,\tau) + \mathrm{A}_{x_2x_1}(\xi,\tau)$$

Let the interference term be denoted

$$J_{x_1x_2}(\xi,\tau) = \mathrm{A}_{x_1x_2}(\xi,\tau) + \mathrm{A}_{x_2x_1}(\xi,\tau)$$

The Fourier relation of equation (32), between the Wigner-Ville and the ambiguity function, proves that

$$I_{x_1x_2} = \iint_{\mathbb{R}^2} J_{x_1x_2}(\xi,\tau) e^{-i2\pi(\xi t + \nu\tau)}\, d\xi\, d\tau$$

For the previous example, in which

$$x_1(t) = e^{i2\pi(\nu_m - \frac{\Delta\nu}{2})t}, \quad \text{and} \quad x_2(t) = e^{i2\pi(\nu_m + \frac{\Delta\nu}{2})t}$$

direct calculation yield

$$\mathrm{A}_{x_1}(\xi,\tau) = \delta(\xi)\, e^{i2\pi(\nu_m - \frac{\Delta\nu}{2})\tau}, \quad \mathrm{A}_{x_1x_2}(\xi,\tau) = \delta(\xi + \Delta\nu)\, e^{i2\pi\nu_m\tau}$$

$$\mathrm{A}_{x_2}(\xi,\tau) = \delta(\xi)\, e^{i2\pi(\nu_m + \frac{\Delta\nu}{2})\tau}, \quad \mathrm{A}_{x_2x_1}(\xi,\tau) = \delta(\xi - \Delta\nu)\, e^{i2\pi\nu_m\tau}$$

Summing these and simplifying yields

$$\mathrm{A}_{x_1+x_2}(\xi,\tau) = \left[2\cos(2\pi\tfrac{\Delta\nu}{2}\tau)\,\delta(\xi) + \delta(\xi+\Delta\nu) + \delta(\xi-\Delta\nu)\right] e^{i2\pi\nu_m\tau}$$

The interference term for this example is

$$J_{x_1x_2}(\xi,\tau) = \left[\delta(\xi+\Delta\nu) + \delta(\xi-\Delta\nu)\right] e^{i2\pi\nu_m\tau}$$

which demonstrates that interferences appear in the ambiguity plane where the Doppler variable ξ is equal to $\pm\Delta\nu$. This quantity represents the frequency difference between the two signal components.

In the context of a dissonance analysis, we could interpret the interferences that appear near the origin as are responsible for beating or "roughness" as it is here that signal components are closest in frequency. However, such an analysis relies on our ability to separate signal and interference components when computing the time-frequency representation. This is crucial, particularly for the ambiguity function in which the true signal components appear near the origin. In that case it is impossible to discern interference terms resulting from interaction among components with small frequency differences, unless we study the interferences terms by themselves. We consider one method of separating signal and interference energy in the following section.

4.5 Energy Separation

Recall the matching pursuit decomposition of section 3.6,

$$x(t) = \sum_{n=0}^{\infty} C(R^n x, g_{\gamma_n}) g_{\gamma_n}(t)$$

Referring to equation (37), we see that the corresponding Wigner-Ville representation is

$$W_x(t,\nu) = \sum_{n=0}^{\infty} \left|C(R^n x, g_{\gamma_n})\right|^2 \mathrm{W}_{g_{\gamma_n}}(t,\nu) + 2\sum_{n=0}^{\infty}\sum_{k=n+1}^{\infty} \mathrm{Re}\left[C(R^n x, g_{\gamma_n}) C^*(R^n x, g_{\gamma_k})\,\mathrm{W}_{g_{\gamma_n} g_{\gamma_k}}(t,\nu)\right] \tag{40}$$

However, in the literature [14], we find a simpler representation of signal energy, defined by

$$E_x(t,\nu) = \sum_{n=0}^{\infty} \left|C(R^n x, g_{\gamma_n})\right|^2 \mathrm{W}_{g_{\gamma_n}}(t,\nu) \tag{41}$$

Thus, only the first term of (40) appears in the definition of E_x, the idea being that this term accounts for the energy of the "true" signal components. Since this is the primary concern in the matching pursuit literature, the definition of signal energy found in such literature does not include interference terms.

Denoting the cross terms of (40) by

$$I_x(t,\nu) = 2\sum_{n=0}^{\infty}\sum_{k=n+1}^{\infty} \mathrm{Re}\left[C(R^n x, g_{\gamma_n}) C^*(R^n x, g_{\gamma_k})\,\mathrm{W}_{g_{\gamma_n} g_{\gamma_k}}(t,\nu)\right]$$

we can write the Wigner-Ville as

$$W_x(t,\nu) = E_x(t,\nu) + I_x(t,\nu)$$

and we call $E_x(t,\nu)$ the *signal energy* and $I_x(t,\nu)$ the *interference energy.*

5 Practical Considerations

5.1 Windowed Fourier Transform

If there is only a single partial in the sound, then the spectrum contains only this one partial. In an ideal setting, the spectrum of a pure sine wave is zero everywhere except at the frequency of the sine wave. However, in practice the FFT of a sine wave is not exactly zero at points other than the true frequency, and there are two different kinds of errors: roundoff (numerical) errors and artifacts ("edge effects"), that cause the representation of a sine wave to "leak" or "smear out" to other frequencies.

To give an example, let T be the period (seconds per cycle) of the sine wave. That is, it takes T seconds to complete one full cycle. Therefore, the wave has frequency $f = 1/T$ cycles per second (Hz). Also, there are 2π radians per cycle, so the wave has frequency $\omega = 2\pi f$ radians per second. Suppose that $f = 220$ Hz and the sample rate is $SR = 44{,}100$ samples per second. Define the *normalized frequency* by

$$f_n = \frac{f}{SR} = \frac{220}{44100} = 0.005$$

and suppose we sample the sine wave at a total of $N = 2^{15}$ points. This provides $N/SR \approx 0.743$ seconds of sample data, during which time the wave completes

$$f \times \frac{N}{SR} = f_n \times N \approx 163.47$$

cycles. Using these observations, and computing the Fourier transform produces figure 6. The peak in the bottom graph correctly indicates that the true frequency is somewhere around 220 Hz, but it is not very precise. The Matlab programs used to generate these figures are given in the appendix at section F.2.

Given a periodic signal, the Fourier transform assumes that the entire set of observations represents an integer number of periods for that signal. When this assumption is valid, concatenating the signal with multiple copies of itself does not introduce any spurious periodicities into the signal. As shown in the top graph of figure 6, this assumption is not always valid. Since the interval over which we sample the signal did not allow for the completion of an integer number of periods, the concatenation – depicted by the dashed extention in the figure – gives a description of the wave as something other than a pure sinusoid.

Contrast the foregoing with what happens when the number of observations in our sample set allows for an integral number of periods. This is easily accomplished when we are free to choose all experimental parameters. Suppose given are the sample rate, $SR = 44100$, and the number of observations $N = 32768$, and suppose also that we merely require the wave closely approximate the given frequency $f = 220$ Hz. Then let

$$\tilde{f} = (SR/N)\lfloor (N/SR)f \rfloor \approx 219.37$$

The second factor, resulting from the "floor" operator $\lfloor \cdot \rfloor$, represents the largest integer no greater than the number of periods (163.47) completed by a 220 Hz wave during a sample time of N/SR seconds. As shown in Figure 7, a concatenation of the waveform – depicted by the dashed continuation at $t = 0.743$ – introduces no spurious frequency components.

From a practical point of view, it is natural to question the utility of a method that depends on our ability to specify the frequencies of the signals under study. Instead, suppose we accept the

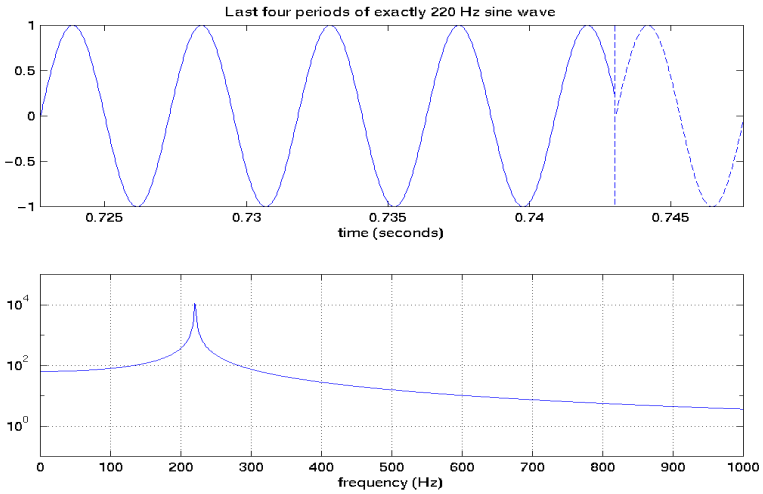

Figure 6: (top) The last four periods of a 220 Hz sine wave sampled over the interval 0 to 0.743 seconds. Since the wave can only complete 163.47 cycles over this time interval, a concatenation of the waveform (employed by the FFT and depicted by the dashed continuation at $t = 0.743$) introduces spurious frequency components into the signal.

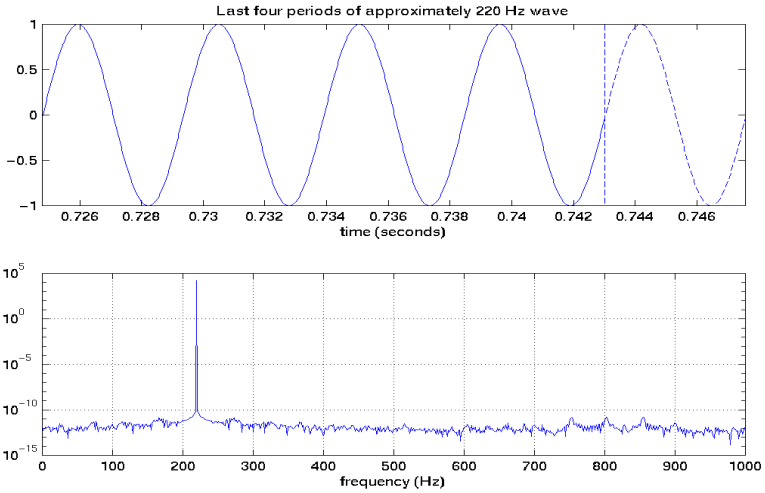

Figure 7: (top) The last four periods of a 219.37 Hz sine wave sampled over the interval 0 to 0.743 seconds. Since the wave completes exactly 163 cycles over this time interval, a concatenation of the waveform (employed by the FFT and depicted by the dashed continuation at $t = 0.743$) introduces no spurious frequency components.

signal frequencies as given and simply adjust the number of samples used in the FFT to match the periods of the existing frequencies. Unfortunately, this is also impractical when analyzing real sounds. For, choosing this length requires knowing the frequencies of the partials, and finding these frequencies is precisely the FFT's raison d'être.

The problem – exposed in Figure 6 – occurs because the (empirical) ends of the signal don't line up; abrupt changes in the waveform cause the spectrum to smear. One way to force the ends to line up is to preprocess the data so that it dies away to zero at both ends. Then, no matter what the underlying periodicity, there will be no abrupt changes in the wave shape.

One popular approach is the *Hamming* window[5]. The effect of the window on a 20 Hz wave is shown in Figure 8. The result of applying the window to a 220 Hz wave and then taking an N-point fast Fourier transform is shown in Figure 9.

The second cause of error when estimating the spectrum of a signal concerns time versus frequency resolution. The FFT measures the frequency content of a signal on a given time interval, of length Δt, over which we assume that the signal is approximately stationary. For real world signals, such as music, the approximation gets worse as Δt increases. That is, the time intervals over which we assume a constant frequency content must be small. Precision in determining *when* various frequencies occur is the result of good or high *time resolution*.

On the other hand, as the time resolution increases, the *frequency resolution* decreases. This trade-off, a result of the well-known *uncertainty principle*, is illustrated by the following scenario. Assume a sample rate of $SR = 44100$ observations per second, and a 4096-point *analysis window* (the interval over which to compute the FFT). This implies that each window examines $\Delta t = N/SR \approx 0.093$ seconds of the signal. In this case, the FFT measures the frequency content of the signal by providing an N-element vector of magnitudes indicating strength of spectral components at (roughly) the frequencies 10.8 Hz, 21.5 Hz, ..., 44100 Hz. The general term in this sequence of frequencies is

$$k \times \frac{SR}{N} = k \times \frac{44100}{4096} \approx k \times 10.77 \text{ Hz}, \qquad k \in \{1, 2, \ldots, N\}$$

As shown by the top graph in Figure 10, the frequency resolution is poor.

Compare this to the same analysis at a lower time resolution, say $\Delta t \approx 0.372$, obtained by setting $N = 16384$. This provides the much clearer picture of the frequency content in the bottom graph of Figure 10. Still, the FFT only indicates magnitudes at frequencies that are integer multiples of $SR/N \approx 2.69$. The presence in the signal of any frequency components that *are not* multiples of 2.69 (e.g. 220) is indicated by an interpolation of the magnitudes at neighboring points (e.g. 218.0 and 220.7) which *are* multiples of 2.69.

5.2 Fourier Based Dissonance Estimation

Consider modeling the signal f as the sum of partials,

$$f(t) = \sum_j p_j(t)$$

with $p_j(t) = a_j(t) \cos \phi_j(t)$ and $\phi_j(t) = \omega_j t + \phi_j$. Measured in Hertz, the instantaneous frequencies of the partials are $f_j = \phi_j'(t)/2\pi$. A measure of the sensory dissonance between two pure sinusoids,

[5]Named after Richard Hamming, this is a single cycle of a scaled and shifted cosine wave. The formula is $h(t) = 0.54 - 0.46 \cos(2\pi t/(N-1))$ for $0 \le t < N$. The Matlab code implementing this technique is given in the appendix at section F.2

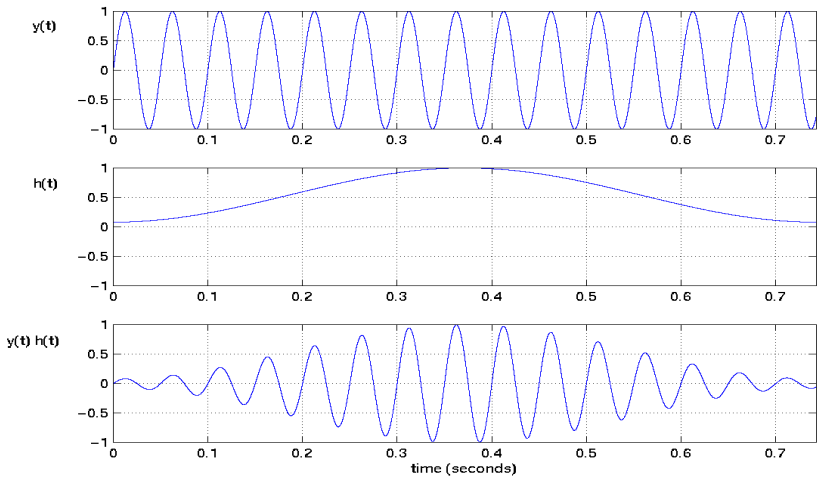

Figure 8: A 20 Hz sine wave (top) and a Hamming window (middle) are multiplied to produce the attenuated sine wave (bottom).

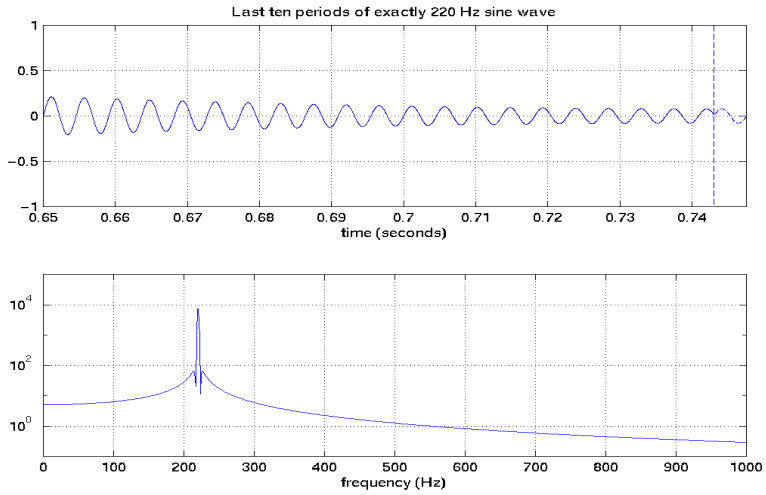

Figure 9: (top) The last 10 periods of a 220 Hz windowed sine wave sampled over the interval 0 to 0.743 seconds. Although the wave does not complete an integral number of periods over this time interval, a concatenation of the waveform introduces less error than before because the wave is attenuated significantly at the point of aperiodicity ($t = 0.743$).

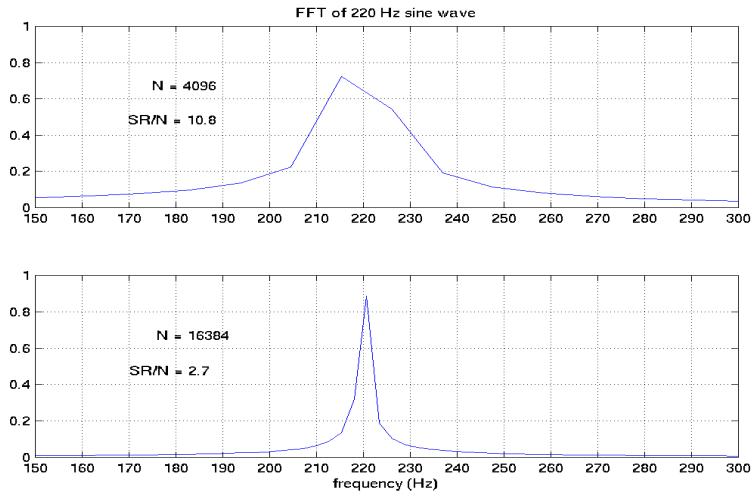

Figure 10: (top) The FFT of a 220 Hz windowed sine wave sampled over the interval 0 to 0.093 seconds. (bottom) The FFT of a 220 Hz windowed sine wave sampled over the interval 0 to 0.372 seconds.

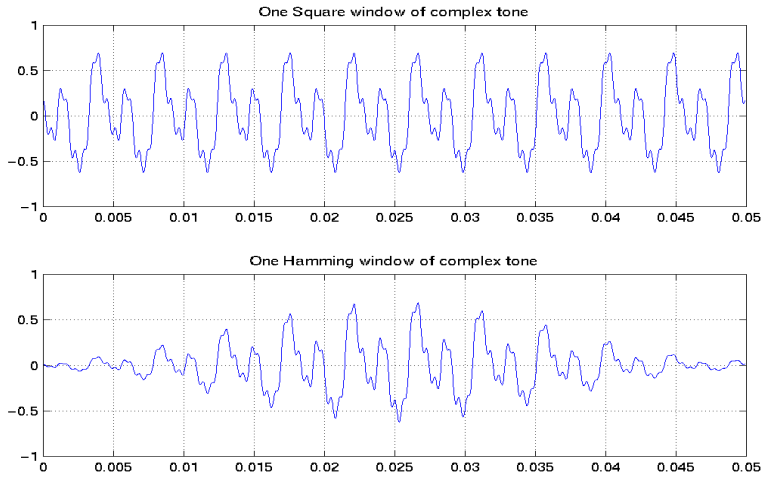

Figure 11: A complex tone with frequencies 220, 440, 880, 1760 Hz and amplitudes .3, .4, .1, .1 (resp.) in a square window (top) and Hamming window (bottom) each of length 0.05 seconds.

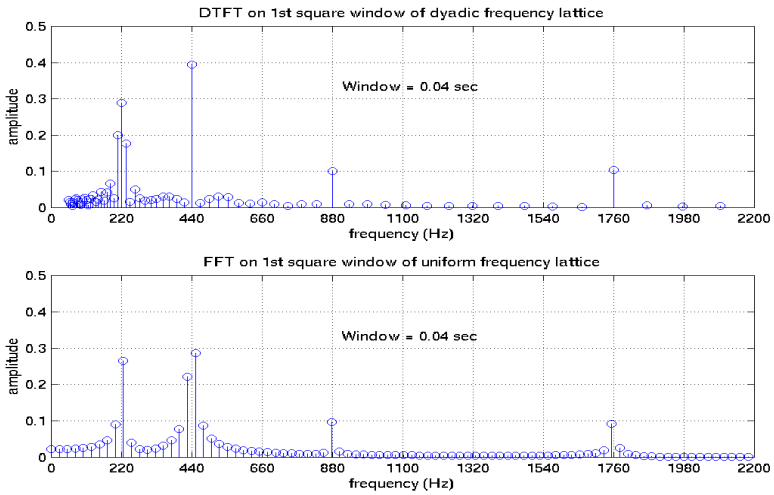

Figure 12: The DTFT (top) and FFT (bottom) of a complex tone with frequencies 220, 440, 880, 1760 Hz and amplitudes .3, .4, .1, .1, (resp.) in a square window of length 0.04 seconds.

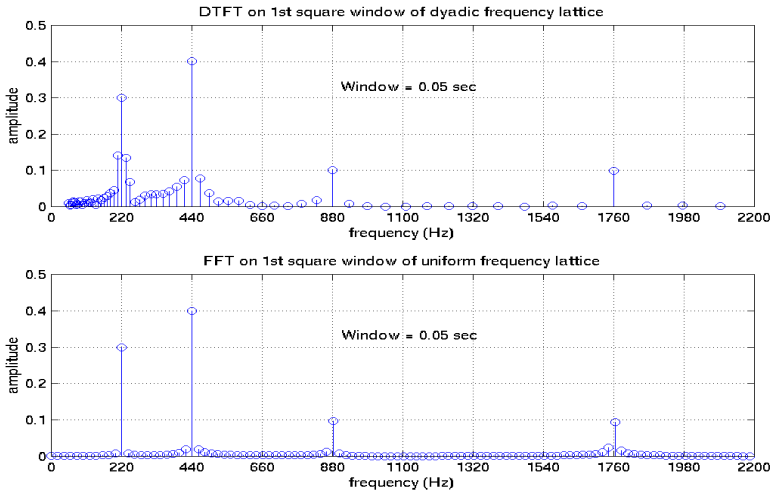

Figure 13: The DTFT (top) and FFT (bottom) of a complex tone with frequencies 220, 440, 880, 1760 Hz and amplitudes .3, .4, .1, .1 (resp.) in a square window of length 0.05 seconds.

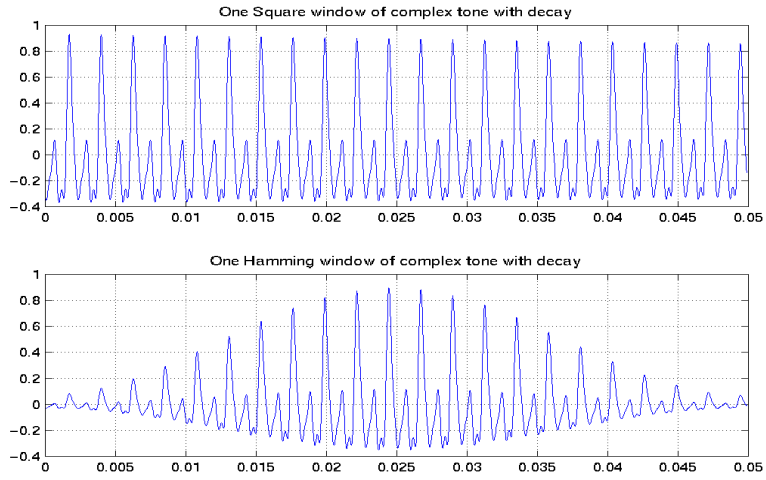

Figure 14: A complex tone with frequencies 440, 880, 1320, 1760, 2200 Hz and amplitudes that start at .3, .4, .1, .1, .08, (resp.), decaying exponentially. (top) a square window and (bottom) a Hamming window each of length 0.05 seconds.

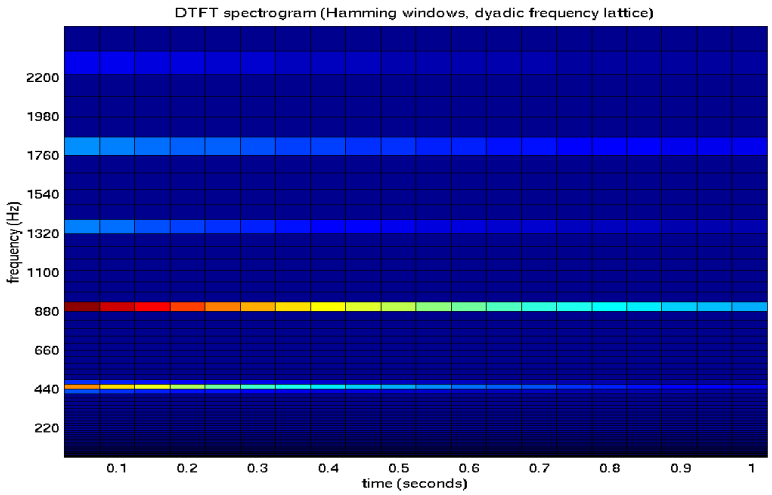

Figure 15: The DTFT of a complex tone with frequencies 440, 880, 1320, 1760, 2200 Hz and amplitudes that start at .3, .4, .1, .1, .08, (resp.), decaying exponentially. The time dimension is divided into 0.05 second Hamming windows. Amplitudes are computed along the frequency axis at the points $55 \cdot 2^{k/12}$ where $k = 0, 1, \ldots, 96$.

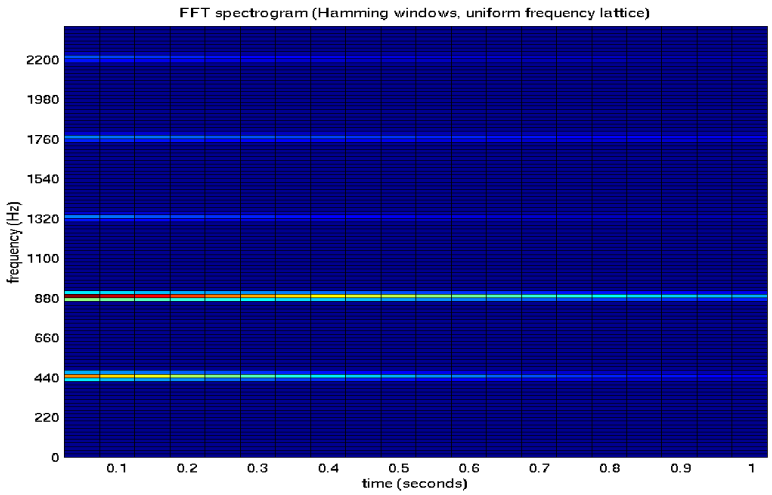

Figure 16: The FFT of a complex tone with frequencies 440, 880, 1320, 1760, 2200 Hz and amplitudes that start at .3, .4, .1, .1, .08, (resp.), decaying exponentially. The time dimension is divided into 0.05 second Hamming windows. Amplitudes are computed along the frequency axis at 2200 evenly spaced frequencies.

p_j and p_k, can be based on the empirical research of Plomp and Levelt [4] discussed earlier.

At time t_0, assume that the values $f_j = \phi_j'(t_0)/2\pi$ and $f_k = \phi_k'(t_0)/2\pi$ are known. Then we define an *instantaneous dissonance* measure between the two partials as follows:

$$D[p_j, p_k](t_0) = a_j(t_0)a_k(t_0)d(f_j, f_k)$$

For the metric $d(\cdot,\cdot)$ we employ the parameterization of the Plomp-Levelt curves given by Sethares [5],

$$d(f_j, f_k) = e^{-b_1 x} - e^{-b_2 x} \tag{42}$$

where $x(t) = |f_j - f_k|/\min\{f_j, f_k\}$. The constants b_1 and b_2 determine the rate at which the dissonance curve rises and falls. Sethares performed a gradient minimization of the squared error between the function (42) and the averaged Plomp and Levelt data. This values $b_1 = 3.5$ and $b_2 = 5.57$ produce the best fit.

Next, we define the *instantaneous dissonance* of the signal f at time t_0 as the sum

$$Df(t_0) = \frac{1}{2}\sum_{j,k} a_j(t_0)a_k(t_0)d(f_j, f_k) \tag{43}$$

Of course, in general we cannot assume that the amplitudes and instantaneous frequencies are known. However, we can approximate these data by assuming them constant over the time support of each windowed Fourier atom $g_{s,u,\xi}$. This assumption permits a measure of dissonance at each window including time t_0 for which $(t_0, \xi(t_0))$ is a ridge point. We take this as our estimate of the instantaneous dissonance over the time window containing t_0. More precisely, let $W_{t_0} = [t_-, t_+]$ be the window containing t_0. At each ridge point $(t_0, \xi_j(t_0))$, with $t_0 \in W_{t_0}$, we estimate

$$\tilde{f}_j = \xi_j(t_0)/2\pi, \qquad \tilde{a}_j = \frac{2|Sf(t_0, \xi_j(t_0)|}{\sqrt{s}|\hat{g}(0)|}$$

We take as our estimated sensory dissonance over the interval $t \in W_{t_0}$

$$\tilde{D}f(t) = \frac{1}{2} \sum_{j,k} \tilde{a}_j \tilde{a}_k d(\tilde{f}_j, \tilde{f}_k) \tag{44}$$

Figure 17 shows the instantaneous dissonance of the signal (17) as t varies over $[0, 1]$. It is computed using (43). Figure 18 shows the estimated dissonance of the same signal. It is computed by taking amplitude and frequency estimates from the ridges in Figure 5 and plugging them into (44). As we can see, the windowed Fourier atoms do not provide a high enough frequency resolution to provide a reliable dissonance measure on the interval $t \in [.3, .4]$. This corresponds to a frequency interval for the linear chirp of about $[418, 484]$ Hz. For a musical signal, this implies that only frequency intervals of more than 2 semitones are distinguishable. One might argue that, for a method to be useful for musical applications, it must be able to discriminate between two components that are one semitone apart.

5.3 Matching Pursuit

This section describes how we implement the atomic decomposition of a signal using the matching pursuit algorithm described in section 3.6.

The Matlab program `MZMP.m`, appearing in section F.5, computes the optimal gabor atoms for representing a given signal. The primary sub-routines it calls are `gaborMZ.m` and `IP.m`, which listed in sections F.7 and F.8, respectively. The `gaborMZ` sub-routine computes individual gabor atoms given a set of parameters. (Section 3.6 describes the parameters that define a gabor atom.) The `IP` sub-routine computes the inner product of two gabor atoms from their given parameters without actually constructing the atoms.

5.4 Energy Separation

This section describes the implementation of the energy separation technique described in section 4.5.

The Matlab program `DEnergy.m` appears in section F.4, and computes the *signal energy* and *interference energy* of a given signal. (The italicized terms are defined in section 4.5.) The first step in this process is a call to the `MZMP` routine, which computes the matching pursuit decomposition of the signal. The output of `MZMP` are the parameters of a collection of gabor atoms which optimally represent the given signal. Next, the Wigner and cross-Wigner transforms of these optimal atoms are computed by calling the sub-routines `WVTrans_AF.m` and `WVTrans_AFC.m`, respectively. Finally, the `DEnergy` routine (optionally) produces a time-frequency display of the signal energy and interference energies.

5.5 Energy Based Dissonance Estimation

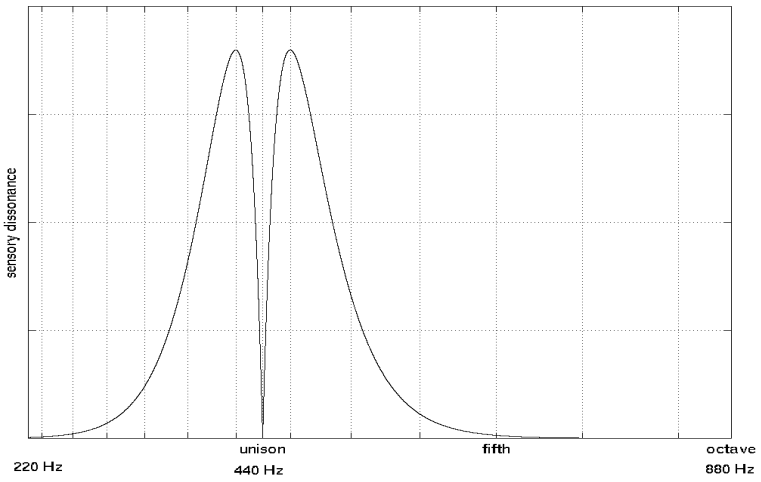

Figure 17: The exact dissonance curve computed with *a priori* knowledge of the amplitudes and frequencies. The horizontal axis delimits time, though the tick marks indicate the frequency of the linear tone.

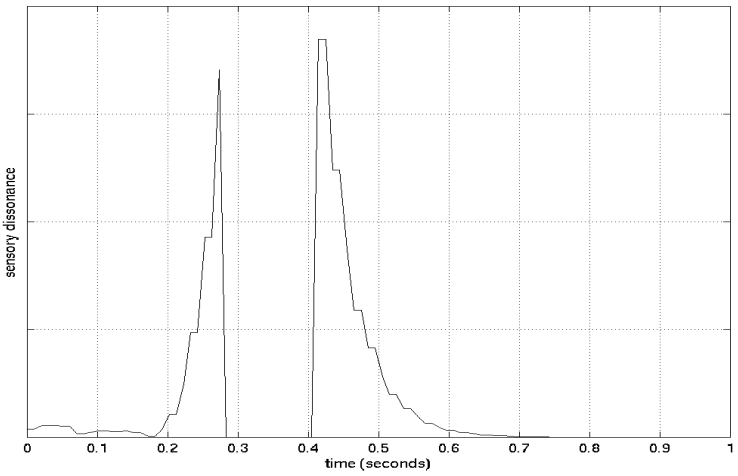

Figure 18: The estimated dissonance curve computed with ridge point estimates of amplitudes and frequencies.

APPENDIX

A Math Background and Notes

A.1 Inner Product Spaces

The signal spaces we consider are the spaces $L^2(a,b)$ and $\ell^2(n_1,n_2)$. On these we can define an *inner product* which assigns a complex number to two signals $x(t)$ and $y(t)$, or $x(n)$ and $y(n)$, respectively. Denoted $\langle x,y\rangle$, the inner product must satisfy the following axioms:

1. $\langle x,y\rangle = \langle y,x\rangle^*$
2. $\langle \alpha x+\beta y,z\rangle = \alpha\langle x,z\rangle + \beta\langle y,z\rangle$
3. $\langle x,x\rangle \geq 0, \qquad \langle x,x\rangle = 0 \Leftrightarrow x = \mathbf{0}$

Here, $\alpha,\beta\in\mathbb{C}$ are scalars, and $\mathbf{0}$ is the null vector.

Some examples of inner products that we employ in the present work are

$$\begin{aligned}
\langle x,y\rangle &= \int_{-\infty}^{\infty} x(t)y^*(t)\,dt, && x,y\in L^2(\mathbb{R})\\
\langle x,y\rangle &= \int_{a}^{b} x(t)y^*(t)\,dt, && x,y\in L^2(a,b)\\
\langle x,y\rangle &= \sum_{n=-\infty}^{\infty} x(n)y^*(n), && x,y\in \ell^2(\mathbb{R})\\
\langle x,y\rangle &= \sum_{n=n_1}^{n_2} x(n)y^*(n), && x,y\in \ell^2(n_1,n_2)
\end{aligned}$$

Each of the foregoing definitions of $\langle x,y\rangle$ corresponds to an associated vector space, appearing on the right, of which x and y are members. Therefore, the meaning of $\langle x,y\rangle$ will often be clear from the space on which it is used.

A.2 Linear Operators

This section reviews some of the operator theory cited in this paper.

The Banach space of bounded linear operators that map the Hilbert space $\mathcal{H}_1$ to the Hilbert space $\mathcal{H}_2$ is denoted $\mathcal{B}(\mathcal{H}_1,\mathcal{H}_2)$. The *norm* of an element $\mathrm{T}\in\mathcal{B}(\mathcal{H}_1,\mathcal{H}_2)$ is

$$\|\,\mathrm{T}\,\| = \sup_{x\in\mathcal{H}_1} \frac{\|\,\mathrm{T}[x]\|_{\mathcal{H}_2}}{\|x\|_{\mathcal{H}_1}} < \infty$$

Two classes of operators specifically worthy of mention are the self-adjoint operators and the orthogonal projection operators.

Self-Adjoint. ([15] Fact 2.2) If $\mathrm{T}\in\mathcal{B}(\mathcal{H},\mathcal{H})$ is *self-adjoint* then

$$\|\,\mathrm{T}\,\| = \sup_{f\in\mathcal{H}} \frac{|\langle f,\mathrm{T}[f]\rangle|}{\|f\|^2}.$$

Orthogonal Projection. ([15] Fact 2.3) Let $\mathcal{X}$ be a subspace of the Hilbert space $\mathcal{H}$ and let $\mathrm{P}_{\mathcal{X}} : \mathcal{H} \mapsto \mathcal{X}$ denote the *orthogonal projection operator* onto $\mathcal{X}$. The operator $\mathrm{P}_{\mathcal{X}} \in \mathcal{B}(\mathcal{H}, \mathcal{H})$ is an orthogonal projection onto $\mathcal{X}$ if and only if $\mathrm{P}_{\mathcal{X}}^2 = \mathrm{P}_{\mathcal{X}}$ and $\mathrm{P}_{\mathcal{X}}$ is self-adjoint.

Following the exposition in Teolis [15], we now state some important properties of linear operators. Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be arbitrary Hilbert spaces with norms $\|\cdot\|_{\mathcal{H}_1}$ and $\|\cdot\|_{\mathcal{H}_2}$ and inner products $\langle\cdot,\cdot\rangle_{\mathcal{H}_1}$ and $\langle\cdot,\cdot\rangle_{\mathcal{H}_2}$, respectively. Let T be an operator that maps a function in $\mathcal{H}_1$ to a function in $\mathcal{H}_2$; that is, $\mathrm{T} : \mathcal{H}_1 \mapsto \mathcal{H}_2$.

1. The *range* of T is $\mathrm{T}(\mathcal{H}_1) \equiv \{\mathrm{T}[f] : f \in \mathcal{H}_1\}$.
2. The *kernel* (or *null space*) of T is $\mathcal{N}(\mathrm{T}) \equiv \{f \in \mathcal{H}_1 : \mathrm{T}[f] = 0\}$.
3. T is *injective* (or *one-to-one*) when $\mathrm{T}[f] = \mathrm{T}\,g$ if and only if $f = g$. If T is a linear operator then T is injective if and only if $\mathcal{N}(\mathrm{T}) = \{0\}$.
4. T is *surjective* or *onto* if $\mathrm{T}(\mathcal{H}_1) = \mathcal{H}_2$.
5. T is *bijective* if it is both injective and surjective.
6. T has an *inverse* $\mathrm{T}^{-1} : \mathcal{H}_2 \mapsto \mathcal{H}_1$ if T is bijective. In this case the inverse of T is defined as $\mathrm{T}^{-1}\,g = f$, where f is such that $\mathrm{T}[f] = g$.
7. T is *continuous* if $x_n \mapsto x$ implies $\mathrm{T}[x]_n \mapsto x$. A linear operator is bounded if and only if it is continuous.
8. The *adjoint* of T is the unique operator $\mathrm{T}^* : \mathcal{H}_2 \mapsto \mathcal{H}_1$ for which $\langle \mathrm{T}[f], g\rangle_{\mathcal{H}_2} = \langle f, \mathrm{T}^*\,g\rangle_{\mathcal{H}_1}$ holds for all $f \in \mathcal{H}_1$ and $g \in \mathcal{H}_2$. T is *self-adjoint* if $\mathrm{T}^* = \mathrm{T}$.
9. $\mathrm{T} \in \mathcal{B}(\mathcal{H}_1, \mathcal{H}_2)$ is a *compact* operator if, for all sequences $\{f_n : \|f_n\| = 1\} \subseteq \mathcal{H}_1$, the sequence $\{\mathrm{T}[f]_n\}$ has a converging subsequence in $\mathcal{H}_2$.
10. T is a *topological isomorphism* if T is bijective, $\mathrm{T} \in \mathcal{B}(\mathcal{H}_1, \mathcal{H}_2)$, and $\mathrm{T}^{-1} \in \mathcal{B}(\mathcal{H}_2, \mathcal{H}_1)$. Thus, both T and T^{-1} are continuous linear operators.
11. T is an *isometry* if for all $f \in \mathcal{H}_1$, $\|\mathrm{T}[f]\|_{\mathcal{H}_2} = \|f\|_{\mathcal{H}_1}$.
12. T is a *unitary* map if it is linear, bijective, and an isometry. If T is unitary then $\mathrm{T}^{-1} = \mathrm{T}^*$.

A.3 Integral Transforms

The integral transform is one of the most important tools in signal theory. The best known example is the Fourier transform, which we describe in section A.5, but there are many other transforms of interest.

The basic idea of an integral representation is to describe a signal $x(t)$ via its density $\hat{x}(t)$ with respect to an arbitrary *kernel*, denoted $\phi(t, s)$. This is done as follows:

$$x(t) = \int_S \hat{x}(s)\phi(t, s)\,ds, \qquad t \in T \tag{45}$$

A *reciprocal kernel*, $\theta(s,t)$, may be found such that the density $\hat{x}(s)$ can be calculated in the form

$$\hat{x}(s) = \int_T \hat{x}(t)\theta(s,t)\,dt, \qquad s \in S \tag{46}$$

Substituting (46) into (45) yields,

$$x(t) = \int_T x(\tau) \int_S \theta(s,\tau)\phi(t,s)\,ds\,d\tau \tag{47}$$

The *Dirac impulse* $\delta(t)$ satisfies such an integral equation:

$$x(t) = \int_{-\infty}^{\infty} \delta(t-\tau)x(\tau)\,d\tau, \qquad x \in L^1(\mathbb{R}) \tag{48}$$

From equation (47) and definition (48), we arrive at the following:

$$0 = \int_T x(\tau) \left(\int_S \theta(s,\tau)\phi(t,s)\,ds - \delta(t-\tau) \right) d\tau$$

which holds for all x if and only if

$$\int_S \theta(s,\tau)\phi(t,s)\,ds = \delta(t-\tau)$$

If, instead, we substitute equation (45) into (46) and proceed as above, *mutatis mutandis*, we arrive at

$$\int_T \phi(t,\xi)\theta(s,t)\,dt = \delta(s-\xi)$$

Self-Reciprocal Kernels. A special category is that of *self-reciprocal kernels*. They correspond to orthonormal bases and satisfy $\phi(t,s) = \theta^*(s,t)$, so that

$$\begin{aligned} \int_S \theta(t,s)\theta^*(s,\tau)\,ds &= \int_S \theta(s,t)\phi(\tau,s)\,ds \\ &= \delta(t-\tau) \end{aligned} \tag{49}$$

Transforms that contain a self-reciprocal kernel are also called *unitary* because they yield $\|\hat{x}\| = \|x\|$, by Parseval's relation, which we cover in the following section.

A.4 Parseval's Relation

Let the signals $x(t)$ and $y(t)$ be square integrable, $x, y \in L^2(T)$. For the densities, let

$$\begin{aligned} \hat{x}(s) &= \int_T x(t)\theta(s,t)\,dt \\ \hat{y}(s) &= \int_T y(t)\theta(s,t)\,dt \end{aligned} \tag{50}$$

where $\theta(s,t)$ is a self-reciprocal kernel satisfying (49). Now consider inner products

$$\begin{aligned} \langle x, y\rangle &= \int_T x(t)y^*(t)\,dt \\ \langle \hat{x}, \hat{y}\rangle &= \int_S \hat{x}(s)\hat{y}^*(s)\,ds \end{aligned} \tag{51}$$

Substituting the densities (50) into the the inner product (51) yields

$$\langle \hat{x}, \hat{y}\rangle = \int_S \int_T \int_T x(\tau)\theta(s,\tau)y^*(t)\theta^*(s,t)\,d\tau\,dt\,ds$$

By self-reciprocity, the foregoing becomes

$$\begin{aligned} \langle \hat{x}, \hat{y}\rangle &= \int_T x(\tau) \int_T y^*(t)\delta(t-\tau)\,dt\,d\tau \\ &= \int_T x(\tau)y^*(\tau)\,d\tau \end{aligned}$$

This proves *Parseval's relation*

$$\langle \hat{x}, \hat{y}\rangle = \langle x, y\rangle$$

For $y(t) = x(t)$, Parseval's relation shows that self-reciprocal kernels are unitary, as claimed in the preceding section:

$$\langle \hat{x}, \hat{x}\rangle = \langle x, x\rangle \qquad \Rightarrow \qquad \|\hat{x}\| = \|x\|$$

A.5 Fourier Transforms

Continuous-Time Fourier Transform. The mapping $\mathcal{F} : L^2(\mathbb{R}) \mapsto L^2(\mathbb{R})$, defined for $x \in L^1(\mathbb{R}) \subset L^2(\mathbb{R})$ by

$$\mathcal{F}[x](\omega) = \hat{x}(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi\omega t}\,dt \tag{52}$$

is called the *continuous-time Fourier transform*. Throughout the paper, we reserve the special notation $X(\omega)$ and $Y(\omega)$ for the Fourier transforms of $x(t)$ and $y(t)$, respectively.

For $x \in L^2(\mathbb{R}) \setminus L^1(\mathbb{R})$, the continuous-time Fourier transform is defined

$$X(\omega) = \lim_{n\to\infty} \int_{-n}^{n} x(t)e^{-i2\pi\omega t}\,dt$$

In the latter case, convergence of the limit to X is in the L^2-sense.

The quantity $X(\omega)$ measures how much oscillation at frequency ω there is in the signal $x(t)$. If $x \in L^1(\mathbb{R})$, the integral in (52) does converge and

$$|X(\omega)| \le \int_{-\infty}^{\infty} |x(t)|dt$$

Therefore X is bounded and it is a continuous function of ω. If X is also integrable, the inverse Fourier transform is

$$x(t) = \int_{-\infty}^{\infty} X(\omega)e^{i2\pi\omega t}\,d\omega \tag{53}$$

which gives the decomposition of the signal $x(t)$ as a weighted sum of sinusoidals $\exp\{i2\pi\omega t\}$ with weights (amplitudes) given by $X(\omega)$. If we define the inner product of the functions x and y to be

$$\langle x, y\rangle = \int_{-\infty}^{\infty} x(t)y^*(t)\,dt$$

then $X(\omega) = \langle x, e^{i2\pi\omega}\rangle$. This operation can be viewed as the projection of x onto $e^{i2\pi\omega}$ since the function $x - Xe^{i2\pi\omega}$ is orthogonal to $e^{i2\pi\omega}$.

To summarize, the inverse Fourier transform (53) represents the reconstruction of $x(t)$ as the sum of its projections onto the basis functions $\{e^{i2\pi\omega t}\}$. Thus, it is simply a more general version of the decomposition appearing in equation (3):

$$x(t) = \sum_{k=1}^{K} a_k \cos(\omega_k t + p_k)$$

The sinusoidal functions $\{e^{i2\pi\omega t}\}$ are useful as building blocks for signals, especially those that do not change quickly over time. However, it is possible to find other *atomic* functions that provide a better basis on which to project signals that have more dramatic stuctural variations. The goal of an atomic decomposition is to provide a recipe for constructing a given function $x \in \mathcal{H}$ out of a set of atomic functions $\{\phi_n\}$. Analysis is restricted to recipes of the form

$$x(t) = \sum c_n \phi_n(t) \tag{54}$$

where $\{c_n\}$ is a countable sequence in $\ell^2(\mathbb{R})$ and the atoms satisfy some fundamental properties. For instance, the basic requirement is that the atoms span the space $\mathcal{H}$ – that is, every element $x \in \mathcal{H}$ can be written as a linear combination of the atoms, as in (54).

Discrete-Time Fourier Transform. Let $x(n)$ be a discretized version of the analog signal considered above. If $x(n)$ is absolutely summable, that is $\sum |x(n)| < \infty$, then its *discrete-time Fourier transform* is given by

$$\mathcal{F}[x_n](\omega) = X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{-i2\pi\omega n} \tag{55}$$

The *inverse discrete-time Fourier transform* of $X(\omega)$ is given by

$$x(n) = \int_{-\infty}^{\infty} X(\omega)e^{i2\pi\omega n}\,d\omega \tag{56}$$

The operator $\mathcal{F}$ transforms a discrete signal $x(n)$ into a complex-valued continuous function $X(\omega)$ of the real variable ω, which is a digital frequency measured in Hertz.

If $x(n)$ is of finite duration, then Matlab can be used to compute $X(\omega)$ numerically at any freuqency ω. The approach is to implement (55) directly. If, in addition, we wish to evaluate $X(\omega)$ at the set of frequencies $\{\omega_1, \ldots, \omega_M\}$, then (55) can be implemented as a simple matrix-vector multiplication. For example, suppose $x(n)$ has N samples between $n_1 \le n \le n_2$. Then (55) can be written as

$$X(\omega_k) = \sum_{\ell=1}^{N} x(n_\ell)e^{-i2\pi\omega_k n_\ell},\ k = 1, \ldots, M \tag{57}$$

When $\{x(n_\ell)\}$ and $\{X(n_\ell)\}$ are arranged as column vectors $\mathbf{x}$ and $\mathbf{X}$, respectively, equation 57 is equivalent to

$$\mathbf{X} = \mathrm{W}\,\mathbf{x}$$

where W is an $M \times N$ matrix given by

$$\mathrm{W} = \left\{ e^{-i2\pi\omega_k n_\ell} : n_1 \le n \le n_2, 1 \le k \le M \right\}$$

If we arrange $\{\omega_k\}$ and $\{n_\ell\}$ as *row* vectors $\mathbf{w}$ and $\mathbf{n}$ repsectively, then

$$\mathrm{W} = \left[\exp\left(-i2\pi \mathbf{w}^t \mathbf{n}\right) \right] \in \mathbb{C}^{M \times N}$$

We will make use of this direct computation of the discrete-time Fourier transform in the experiments of section 5.

Unitarity. The Parseval relation (A.4) shows that the Fourier transform is an isometry of $L^2(\mathbb{R})$. Thus, the inner product of two signals remains the same, no matter if we represent it in the time or frequency domain: $\langle X, Y \rangle = \langle x, y \rangle$. It follows that the temporal and spectral energy densities, respectively $|x(t)|^2$ and $|X(\omega)|^2$, satisfy the energy conservation equations

$$\begin{aligned} \|x\|^2 &= \int_{-\infty}^{\infty} |x(t)|^2 \, dt \\ &= \int_{-\infty}^{\infty} |X(\omega)|^2 \, d\omega = \|X\|^2 \end{aligned}$$

Moyal's formula [16] proves that the Wigner transform is also unitary, an thus yields similar energy conservation properties.

Theorem A.1 (Moyal[6]) For any x and y in $L^2(\mathbb{R})$

$$\begin{aligned} |\langle x, y \rangle|^2 &= \left| \int_{-\infty}^{\infty} x(t) y^*(t) dt \right|^2 \\ &= \iint \mathrm{W}_x(t, \nu)\, \mathrm{W}_y(t, \nu)\, dt\, d\nu \end{aligned} \tag{58}$$

A.6 Cohen's Class

Above we claimed that time-frequency energy densities, such as the spectrogram and scalogram, are time-frequency averagings of the Wigner transform. To see this, reconsider the family $\{\phi_\gamma\}$ of time-frequency atoms from section 3, where for any (u, ω) there exists a unique atom $\phi_{\gamma(u,\omega)}$ centered in time-frequency at (u, ω). The corresponding linear time-frequency transform of x is

$$\mathrm{T}[x](u, \omega) = \langle x, \phi_{\gamma(u,\omega)} \rangle$$

The resulting time-frequency energy density is

$$\begin{aligned} E_{T[x]}(u, \omega) &= \left| \langle x, \phi_{\gamma(u,\omega)} \rangle \right|^2 \\ &= \left| \int_{-\infty}^{\infty} x(t) \phi^*_{\gamma(u,\omega)}(t)\, dt \right|^2 \end{aligned}$$

Moyal's formula (58) proves that this energy density is a time-frequency averaging of the Wigner-Ville transform:

$$E_{T[x]}(u,\omega) = \iint \mathrm{W}_x(t,\nu)\,\mathrm{W}_{\phi_{\gamma(u,\omega)}}(t,\nu)\,dt\,d\nu$$

The smoothing kernel is the Wigner-Ville transform of the atoms

$$\theta(u,t,\omega,\nu) = \mathrm{W}_{\phi_{\gamma(u,\omega)}}(t,\nu)$$

The loss of time-frequency resolution depends on the spread of the transform $\mathrm{W}_{\phi_{\gamma(u,\omega)}}(t,\nu)$ in the neighborhood of (u,ω). To summarize, positive time-frequency transforms totally remove the interference terms, but produce a loss of information.

B Perception of Complex Tones

This section contains notes on the perception of complex tones. In particular, we focus on two concepts that are related to the ideas in the paper, but which are not described therein, as they are not immediately relevant. This material is included here because it might be useful in future work.

B.1 Tonal Fission and Fusion

In this section are some notes on tonal fusion. First are excerpts from Sethares [5] (p. 25, 26), and then some brief notes from Hartmann [6].

Almost all musical sounds consist of a great many partials, whether they are harmonically related or not. Using techniques such as selective damping and the selective excitation models, it is possible to learn to "hear out" these partials, to directly perceive the spectrum of the sound. This kind of listening is called *analytic* listening, as compared to *holistic* listening in which the partials fuse together into one perceptual entity.

Consider the closing chord of a string quartet. At one extreme, is the fully analytic listener that "hears out" a large number of individual partials. At the other extreme is the fully holistic listener who hears the chord as one grand "tone," all four instruments fusing into a single rich and complex sonic texture. Typical listening lies somewhere in between: the partials of each instrument fuse, but the instruments remain individually perceptible, each with its own pitch, loudness, vibrato, etc. What physical cluesmake this remarkable feat of perception possible?

When listeners are presented with clusters of partials and asked how many distinct voices, notes, etc. they hear, various features of the presentation reliably encourage tonal fusion. Some of these features are whether the partials:

1. begin at the same time (attack synchrony),
2. have similar envelopes (amplitudes change similarly over time),
3. are harmonically related, or
4. have the same vibrato rate.

Almost any common feature of a subgroup of partials helps them to be perceived together.

Hartmann [6] (p. 134) states that the most important mediator of perceptual segregation and integration is the onset time of the tones, which he calls *onset synchronicity* (cf. attack syncrony of Sethares). If two sets of partials have different onset times, then the auditory system segregates them as different entities.

A study by Rasch (1978) indicates that 30ms advanced onset leads to 40dB increase in level perception and allows the listener to segregate while remaining unaware of the 30ms asynchrony. However, research by Risset, Matthews (1969) and Beauchamp (1975) shows that the harmonics of high frequencies lag those of low frequencies by greater than 30ms[7]

B.2 Spectral Dominance

A periodic complex tone has a pitch that is equal to, or slightly less than, the frequency of the fundamental. Assuming the various overtones of a complex tone fuse together to create an overall pitch, just how these overtones are combined is a question of great interest. In particular, if a complex tone is composed of pure sinusoidal partials, each having a given frequency and amplitude, which of these partials will be more pronounced and make a greater contribution to the determination of the fundamental (low) pitch of the complex tone? A promising experimental approach is to mistune one or more partials and study the effect on the synthesized low pitch. Experiments by Ritsma (1967) and Moore, Glasberg, and Peters (1985) led to the conclusion that the partials that are most important in determining the low pitch are those that are resolved by the auditory periphery. Retsma concluded that for fundamental frequencies between 100 and 400 Hz, partials three, four, and five are dominant. Terhardt (1974) proposed a dominant frequency range centered on 700 Hz, instead of particular harmonic numbers. Moore *et al.* concluded that the dominant partials were among the first six, but individual differences prevented them from being more specific.

The observed importance of the resolved harmonics supports the idea that pitch perception takes place at high levels where excitations from different peripheral channels are recombined. Pattern matching or template fitting models focus on this idea. The model studied by Goldstein (1973) derives the low pitch from a pattern match to the frequencies of the higher partials.

C Sound Examples

This section presents some notes on the music, used in the experiments of section 5.

C.1 *Metamorphosis* (1988), by Philip Glass[8]

Philip Glass (b. 1937) may be the most wide-ranging of the minimalist composers. He plays keyboards in a crossover group, the Philip Glass Players, which has appeared in concert halls and art museums, at rock clubs and jazz festivals. He has written music for some remarkable art films; he collaborates with the innovative multimedia artist Robert Wilson. The easy-to-play piano piece that this paper examines makes a much quieter statement than the famous Glass pieces such as his operas and ensemble numbers written for his group. Nevertheless, *Metamorphosis* is typical of the composer's work in its constructive principles and overall effect.

[7] A diagram of this phenomena is in the comp book at (00.07.10).

[8] This brief presentation is based on that of Kerman and Tomlinson [17]

Metamorphsis consists of five very similar movements of approximately the same length. In *Metamorphosis 1*, three brief elements alternate, without transitions of any kind to link them. The first element, **a**, announces four chords in a slow dotted rhythm; the second, **b**, introduces gentle motion, oscillating quietly between two notes of a simple chord. This music is utterly uneventful, except that the bass comes in halfway through the four-measure span. The third element, **c**, is a fragment of melody – repetitive, plaintive, built over harmonies similar to those of **a**, and with a fluid accompaniment continuing from **b**. The dynamic never rises above mezzo forte.

This simple plan for *Metamorphosis 1* can be represented

$$||: \mathbf{a}\ \mathbf{b}\ \mathbf{a}\ \mathbf{b}\ \mathbf{c}\ \mathbf{b}\ \mathbf{c}\ \mathbf{b}\ \mathbf{c}\ \mathbf{b} :|| \ \mathbf{a'}\ \mathbf{a'}\ \mathbf{a'}\ \mathbf{b}\ \mathbf{b}\ \mathbf{b} \text{ (repeated three times)}$$

where $\mathbf{a'}$ is an extension of the four-chord series to five – a telling change in these inert surroundings. Notice that although the ending of the chord series features rich and rather melancholy chords, the way the three elements are juxtaposed somehow leaches all the expressivity out of them. One waits fascinated for that inevitable bass entry in b.

D A Note on Numerical Precision

D.1 Numerical Support of Gabor Atoms

Here we remark on some practical considerations which we may want to account for in future (more numerically precise) constructions of the Gabor dictionary.

Suppose we begin with a general gaussian function

$$g(t) = e^{-\alpha t^2}, \qquad t \in \mathbb{R},\ \alpha > 0$$

Then we translate g in time by p, scale it by s, and modulate it by f Hz, yielding

$$g_\gamma(t) = Kg\left(\frac{t-p}{s}\right)e^{i2\pi ft} = Ke^{-\alpha\left(\frac{t-p}{s}\right)^2}e^{i2\pi ft}$$

where $\gamma = (p, s, f)$ denotes the paramter vector and K is chosen so that $\|g_\gamma\| = 1$.

Clearly $g > 0$ for all t, and $g(t) \to 0$ as $t \to \pm\infty$. Thus, the *theoretical support* of g_γ is the entire real line (minus the zeros resulting from the frequency modulation). However, for practical applications in which we are restricted to finite precision arithmetic, it is important to consider the *numerical support* of g_γ, which is necessarily a finite subset of $\mathbb{R}$. In other words, we must consider for what values of t is it the case that $g_\gamma(t) \geq \epsilon_m$. Here ϵ_m is the *machine epsilon*, defined as the smallest positive number attainable by the computer. For example, on our Pentium II hardware, the Matlab function `eps` returns

$$\epsilon_m \approx 2.22 \times 10^{-16}$$

Clearly, when t is such that $|g_\gamma(t)| < \epsilon_m$, the computer perceives $g_\gamma(t)$ as no different from 0, and we must assume that the function is not numerically supported on these values of t.

Let us now find the numerical support of g_γ, for $p = 0$; that is, for $\gamma = (0, s, f)$.

$$
\begin{aligned}
|g_\gamma(t)| \geq \epsilon_m \quad &\Leftrightarrow \quad |Ke^{-\alpha\left(\frac{t}{s}\right)^2} e^{i2\pi ft}| \geq \epsilon_m && (59)\\
&\Leftrightarrow \quad Ke^{-\alpha\left(\frac{t}{s}\right)^2} \geq \epsilon_m && (60)\\
&\Leftrightarrow \quad -\alpha\left(\frac{t}{s}\right)^2 \geq \log\epsilon_m - \log K\\
&\Leftrightarrow \quad \left(\frac{t}{s}\right)^2 \leq \frac{\log K - \log\epsilon_m}{\alpha}\\
&\Leftrightarrow \quad |t| \leq s\left(\frac{\log K - \log\epsilon_m}{\alpha}\right)^{1/2} && (61)
\end{aligned}
$$

Since ϵ_m is a small positive number, $\log K - \log\epsilon_m$ is positive. Thus, the square root in the last inequality of (61) results in a positive real number, and defines the numerical support of g_γ.

It is convenient to choose α such that the numerical support of g_γ is $[-s/2, s/2]$. However, such an α is difficult to acheive analytically because the normalization constant K depends on α and s. So, we proceed by deriving K and α analytically as far as possible, and then we consider a numerical optimization that will yield Gabor atoms having the desired numerical support.

We seek K such that $\|g_\gamma\| = 1$, where

$$g_\gamma(t) = Ke^{-\alpha\left(\frac{t}{s}\right)^2} e^{i2\pi ft}, \qquad t \in \mathbb{R},\ \alpha > 0$$

and

$$\|g_\gamma\|^2 = \int_{-\infty}^{\infty} g_\gamma(t) g_\gamma^*(t)\, dt = K^2 \int_{-\infty}^{\infty} e^{-2\alpha\left(\frac{t}{s}\right)^2} dt$$

Such a K is derived as follows:

$$
\begin{aligned}
1 = \|g_\gamma\|^2 &= K^2 \int_{-\infty}^{\infty} e^{-2\alpha\left(\frac{t}{s}\right)^2} dt\\
\Leftrightarrow \quad K^{-2} &= \int_{-\infty}^{\infty} e^{-2\alpha\left(\frac{t}{s}\right)^2} dt\\
\Leftrightarrow \quad K^{-2} &= \int_{-\infty}^{\infty} \exp\left\{-\frac{t^2}{2(s/2\sqrt{\alpha})^2}\right\} dt
\end{aligned}
$$

Letting $\sigma = \frac{s}{2\sqrt{\alpha}}$, the foregoing is equivalent to

$$K^{-2} = \int_{-\infty}^{\infty} e^{\frac{-t^2}{2\sigma^2}}\, dt = \sigma\sqrt{2\pi}$$

The final equality obtains from the fact that a gaussian density integrates to 1. This shows that the value of K resulting in atoms of unit norm is

$$K = (\sigma\sqrt{2\pi})^{-1/2} = \left(\frac{2\alpha}{\pi s^2}\right)^{1/4}$$

Inserting this expression into equation (61), which defines the numerical support, we have

$$\frac{\log K - \log\epsilon_m}{\alpha} = \frac{1}{4}\left(\frac{\log\alpha - 2\log s - 4\log\epsilon_m + \log(2/\pi)}{\alpha}\right) \qquad (62)$$

Recall from (61) that we seek a value of α such that the right hand side of equation (62) is equal to 1/4. This occurs when

$$\alpha = \log \alpha - 2 \log s - 4 \log \epsilon_m + \log(2/\pi)$$

which is nonlinear in α. Therefore, for given values of s, we select numerically optimal values of α by minimizing the function

$$L_s(\alpha) = |\alpha - \log \alpha + 2 \log s + 4 \log \epsilon_m - \log(2/\pi)|$$

The Matlab routine `fminbnd.m` accomplishes this effectively. We use it as follows:

```
>> fn = inline('abs(x-log(x)+4*log(eps)+2*log(P1)-log(2/pi))',1);
>> alpha = fminbnd(fn,1,200,optimset('Display','off'),S);
```

E Description of Matlab Routines

First briefly state what each routine does, then we will give a more detailed description of the main routines below. The file `beats.m` constructs and returns an example signal. For example,

```
>> signal = beats(K);
```

constructs a signal of length $N = 2^{K+1}$. Specifying signal length with the argument K – representing one less than the integer power of 2 – has certain advantages that become clear later. Briefly, no matter what value is given for K, the signal length will always be a power of 2, which is useful for certain algorithms. Also, knowing the relation $N = 2^{K+1}$, and having the pair N, K at our disposal, facilitates handling the dictionary of Gabor atoms; we use atoms with scales of size 2^j where j ranges from 1 to K.

The file `Dictionary.m` contains code for computing and storing the parameters of all atoms in a Gabor dictionary, as well as storing the correlations of these atoms with the given signal. (If we know the signal length in advance, it is possible to precompute all the atoms and store them to disk. However, depending on the speed of our algorithms for computing atoms on the fly, this may not be the most efficient approach, since it would require retrieval of atoms from slow memory).

Here is an example call to `Dictionary`, with the analytic signal specified in the row vector `signal`:

```
>> [P,C]=Dictionary(signal,DEBUG,DISP);
```

Now that we have a dictionary of Gabor atoms, and the correlations of each atom with the signal, we can call the (Mallat/Zhang) Matching Pursuit routine, `MZMP`:

```
>> [MaxP, MaxC]=MZMP(signal,ITERS,TOL,P,C,DEBUG,DISP); % run MP on signal
```

In the argument list appears `ITERS`, the maximum number of iterations, which is equivalent to the maximum number of atoms with which to approximate the signal; `TOL` is the error tolerance were willing to accept; `P` is the matrix of parameters for all the atoms in the dictionary; `C` is a vector of the signal-atom correlations, for all the atoms in the dictionary. The routine returns `MaxP` which is a matrix of parameters defining the atoms having the best correlation to the signal (or its remainder at each step of the matching pursuit), and `MaxC`, which are those maximum correlation values.

Finally, the routine `DEnergy.m` takes the results from `MZMP` and computes the Wigner energy and interferences. An example call of this routine is:

```
>> [WVE, WVI]=DEnergy(signal,MaxP,MaxC,DEBUG,DISP);  % find WV
```

E.1 Energy Separation

```
function [WVE,WVI,P_DEnergy,C_DEnergy] = DEnergy(sig,DEBUG,DISP,P,C);
```

The `DEnergy.m` function computes energy and interference by calling the matching pursuit routine MZMP, and then the Wigner transform and cross Wigner transform routines, WVTrans_AF and WVTrans_AFC.

```
% Inputs
%   sig       1-dimensional signal of interest
```

```
%   DEBUG     (default 1) DEBUG=0 => do not print extra debugging info
%   DISP      (default 1) DISP=0 => do not display plot of results
%   P         optional input array of atom parameters from a prior run
%   C         optional input vector of atom correlations from a prior
%             run
%
% Outputs
%   WVE       complex-valued matrix representing the sum of the
%             Wigner transforms of each atom in the collection of atoms
%             used to represent the signal. Rows correspond to
%             frequencies and columns corresponding to times.
%   WVI       Same as WVE except that *cross* Wigner transforms are
%             summed.
%   P_DEnergy optionally stored array of atom parameters
%   C_DEnergy optionally stored vector of atom correlations
%
% Usage
%  >> pause off; [WVE,WVI,P_DEnergy,C_DEnergy]=DEnergy;
%  >> [WVE,WVI] = DEnergy(sig,1,1,P_DEnergy,C_DEnergy);
%
% See also:  MZMP, gaborMZ, WVTrans_AF, WVTrans_AFC
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 July 20
% Copyright (c) 2001, William DeMeo

if nargin < 1,
  disp('DEnergy: no signal input... proceding with synthetic signal demo.');
  K=8; N=pow2(K+1);       % signal length
  %%% synthetic signal params (signal should really be an input) %%%%
  f1 = .1; f2 = 2*f1; f3 = 3*f1; %f1 = 440/N; f2 = 2*f1; f3 = 3*f1;
  sig=atoms(N,[1,f1,pow2(K-2),.75; 1,f2,pow2(K-2),.25; ...
    pow2(K),f2,pow2(K-2),1; 3*pow2(K-1),f3,pow2(K-2),.5]);
else,
  N=length(sig); K=nextpow2(N)-1;
end;
if nargin < 2, DEBUG=1; end;
if nargin < 3, DISP=1; end;

% Matching Pursuit parameters
MAXITERS=20;
TOLERANCE=0.05;

if nargin < 5,
  if(DEBUG),disp('DEnergy: computing new atoms with MZMP...');end;
  %%% CALL MATCHING PURSUIT FUNCTION %%%
  [P,C] = MZMP(sig,MAXITERS,TOLERANCE,DEBUG);
end;
% store a local copy of parameter/correlation book of atoms
P_DEnergy = P; C_DEnergy = C;  % (and return it if requested)
```

```
% P is at most a size MAXITERSx4 array; C is at most 1xMAXITERS
M = size(P,1);     % M = number of atoms used in signal representation
WVE = zeros(N,N);
WVI = zeros(N,N);

if(DEBUG),disp('DEnergy: computing energy & interference...');end;
for n=1:M-1,
  [g1,Ks] = gaborMZ(K,P(n,1),P(n,2),P(n,3));
  WV = WVTrans_AF(g1,0);
  WVE = WVE + (abs(C(n))^2)*WV;
  for m=n+1:M,
    [g2,Ks] = gaborMZ(K,P(m,1),P(m,2),P(m,3));
    WV = WVTrans_AFC(g1,g2,0);
    WVI = WVI + 2*real(C(n)*conj(C(m))*WV);
  end;
  if(DEBUG),disp(sprintf('DEnergy: finished atom %d of %d',n,M));end;
end;

[g1,Ks] = gaborMZ(K,P(M,1),P(M,2),P(M,3));
WV = WVTrans_AF(g1,0);
WVE = WVE + (abs(C(M))^2)*WV;

if(DISP),
  figure(1);clf;
  absE = real(WVE);
  tfmax = max(max(absE));
  tfmin = min(min(absE));
  colormap(1-gray(256));
  image(linspace(0,1,N),linspace(0,1,N),256*(absE-tfmin)/(tfmax-tfmin));
  axis('xy'); title('Wigner Energy'); xlabel('Time'); ylabel('Frequency')

  figure(2);clf;
  absI = real(WVI);
  tfmax = max(max(absI)); tfmin = min(min(absI));
  colormap(1-gray(256));
  image(linspace(0,1,N),linspace(0,1,N),256*(absI-tfmin)/(tfmax-tfmin));
  axis('xy'); title('Wigner Interference'); xlabel('Time'); ylabel('Frequency');

  figure(3);clf;
  abstf = real(WVE+WVI);
  tfmax = max(max(abstf)); tfmin = min(min(abstf));
  colormap(1-gray(256));
  image(linspace(0,1,N),linspace(0,1,N),256*(abstf-tfmin)/(tfmax-tfmin));
  axis('xy'); title('Wigner Transform'); xlabel('Time'); ylabel('Frequency')
end;

%% end of file: DEnergy.m
```

E.2 Matching Pursuit

```
function [MaxP,MaxC,P_MZMP,C_MZMP]=MZMP(sig,MAXITERS,TOLERANCE,DEBUG,P,C)
```

```
% MZMP.m
% Matlab function for Mallat/Zhang Matching Pursuit (MZMP) algorithm.
%
% Inputs
%    sig       signal (or part thereof) with which to correlate atoms;
%              signal length must be a power of 2
%    MAXITERS  (default 20) stop after MAXITERS iterations, even
%              if TOLERANCE is not reached.
%    TOLERANCE (default 0.05) stop after relative error is less
%              than TOLERANCE (i.e. |Rx|/|x| < TOLERANCE)
%    DEBUG     (default 0) DEBUG=1 ==> print extra debugging information
%    P         optionally input NAtomsx4 array of all atom parameters
%    C         optionally input 1xNAtoms array of all inner products
%
% Outputs
%    MaxP   MAXITERSx4 array of atom parameters. MaxP(i,:) are the 4
%           parameters specifying the the atoms having correlation
%           MaxC(i) with the signal remainder at step i.
%    MaxC   1xMAXITERS vector where MaxC(i) is the correlation of the
%           signal remainder with max correl atom at iteration i.
%    P_MZMP optionally stored NAtomsx4 array of all atom parameters
%    C_MZMP optionally stored 1xNAtoms array of all inner products
%
% Notes
%    The total number of atoms in the dictionary is computed as follows:
%     (K+2)*4*N = (K+2)*2^(K+3) = 2^(K+3)    Diracs
%                               + 2^(K+3)*K  Gabor atoms
%                               + 2^(K+3)    Complex exponentials
%
%    This version does not consider whether the signal is real or
%    complex and, therefore, the resulting decomposition comprises
%    complex atoms. See MZRealMP for special handling of real signals.
%
% Other Notes
%    Now that this program has been tested and seems to work, all
%    major debugging conditionals have been removed in the interest of
%    efficiency.  See MZMP_DEBUG.m
%
% See also: gaborMZ.m, IP.m, DisplayParams.m
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 May 4
% Copyright (c) 2001, William DeMeo

if nargin < 1,
  if(DEBUG),
    disp('MZMP: no signal input...');
    disp('MZMP: ...proceding with synthetic signal demo.');
  end;
  K=8; N=pow2(K+1);       % signal length
  %%% synthetic signal params (signal should really be an input) %%%%
```

```
  f1 = .1; f2 = 2*f1; f3 = 3*f1; %f1 = 440/N; f2 = 2*f1; f3 = 3*f1;
  sig=atoms(N,[1,f1,pow2(K-2),.75; 1,f2,pow2(K-2),.25; ...
    pow2(K),f2,pow2(K-2),1; 3*pow2(K-1),f3,pow2(K-2),.5]);
else,
  N=length(sig); K=nextpow2(N)-1;
end;

if nargin < 2, MAXITERS = 20; end;     % max number of iterations
if nargin < 3, TOLERANCE = 0.05; end;  % stop when |Rx|/|x| < TOLERANCE
if nargin < 4, DEBUG=0; end;

x=conj(sig(:)); x = x'; % signal is stored as ROW vector x
if(isreal(x)),
  error('use MZMPReal.m for real signals');
end;
x = hilbert(x);         % make signal analytic

pK3 = pow2(K+3);
OPT=3;                  % optimization coefficient
Nopts = pow2(OPT)-1;    % number of frequencies over which to optimize
                        % e.g. 3 => optimize over 7 frequencies
NAtoms = (K+2)*pK3;     % number of atoms in dictionary
% Ex. of (K,NAtoms) pairs: (9,45056) (10,98304) (11,212992) (12,458752)

MaxP=zeros(MAXITERS,4); % store all max parameters
MaxC=zeros(1,MAXITERS); % store all max correl values
RxoxE=zeros(1,MAXITERS);% est. norm of remainder at each iter.

DEBUG=1;  % DEBUG=1 ==> print basic tests and messages
          % DEBUG=2 ==> extra msgs. (now appears only in MZMPoptDEBUG.m)
if(DEBUG),
  disp('pause is on, hit any key...'); pause;
  disp('============================');
  disp('MZMP: STEP I. Initialization');
  disp(sprintf('MZMP: %d atoms in dictionary...',NAtoms));
end;

if nargin < 6,
  P = zeros(NAtoms,4);    %%% Parameters (P), Correlations (C) for:
  C = zeros(1,NAtoms);    %   Diracs         ji = 0,
                          %   Gabor atoms    0 < j < K+1,
                          %   Fourier basis  ji = K+1
  indx = 0;
  for j0=0:K+1, pj0 = pow2(j0);
    for k1=(1:2*pj0), fn = (k1-1)/(2*pj0);
      for u1 = (1:2*N/pj0), u=(u1-1)*pj0/2;
        indx = indx + 1; % == j0*pK3 + (k1-1)*pow2(K-j0+2) + u1;
        [g, Ks] = gaborMZ(K,j0,u,fn);
        C(indx) = x*g';  % same: C(i)=sum(x.*conj(g)); BECAUSE PRIMED!
        P(indx,:) = [j0 u fn Ks];
      end;
```

```
    end;
    if(DEBUG),disp(sprintf('MZMP:       ...finished index %d',indx));end;
  end;
else,
  disp('MZMP: ...using previously computed atoms.');
end;
P_MZMP = P; C_MZMP = C;

[maxC,maxCi]=max(abs(C));  % get max abs corr. value and index
MaxC(1) = C(maxCi);        % not the same as maxC (which is an abs value)
MaxP(1,:) = P(maxCi,:);    %[j0 u fn Ks];

if(DEBUG),
  DisplayParams('MZMP: (before opt) MaxP(1,:) = ',MaxP(1,:),4);
  DisplayParams('MZMP:              MaxC(1) = ',MaxC(1),1);
end;

%%% I.b. OPTIMIZE %%% (only for non-Dirac, modulated atoms)
if(P(maxCi,1)>0 & P(maxCi,3)>0),          % non-Dirac atoms w/ pos. freq.
  j0 = P(maxCi,1); pj0 = pow2(j0); u0 = P(maxCi,2); f0 = P(maxCi,3);
  fopt = f0 + (-OPT:OPT)/(pow2(OPT)*pj0); % 2^OPT - 1 freqs around f0
  if(DEBUG & find(fopt > 1)),
    disp('WARNING: frequency values must be no greater than 1');
  end;
  [g, Ks] = gaborMZ(K,j0,u0,fopt);   % Nopts rows, one atom per row
  Copt = x*g';                       % Nopts x 1 col. of ips
  [maxCopt,maxCoptI]=max(abs(Copt)); % get opt max corr value and index
  if(maxCopt > maxC),                % use new opt freq
    MaxC(1)=Copt(maxCoptI); MaxP(1,3)=fopt(maxCoptI); % MaxP(1,4)=Ks;
    if(DEBUG),
      disp(sprintf('MZMP: (compare) |MaxC| = %g, |MaxCopt| = %g',...
          maxC,maxCopt));
      DisplayParams('MZMP: (after opt)  MaxP(1,:) = ',MaxP(1,:),4);
      DisplayParams('MZMP:              MaxC(1) = ',MaxC(1),1);
    end;
  end;
end;

%% Relative norm of estimated remainder %%
normx = norm(x);  % normx = sqrt(x*x');
normDiff = normx^2 - abs(MaxC(1))^2;
normRx = sqrt(abs(normDiff));  RxoxE(1) = normRx/normx;
if(DEBUG),
  if(normDiff < 0),
    disp('WARNING: |correlation| > |signal|');
    disp(sprintf('MZMP: %g = abs(MaxC(1)) > normx = %g',...
        abs(MaxC(1)),normx));
  end;
  DisplayParams('MZMP: est. |Rx|/|x| = ',RxoxE(1),1);
  disp('MZMP:        ...STEP I. complete.');
  disp('---------------------------------');
```

```
  disp('MZMP: Step II. Update and Iterate');
end; n=0;
while n < MAXITERS,
  n = n+1;
  if(DEBUG), disp(sprintf('MZMP: ~~~~~~~ iteration %d ~~~~~~~',n)); end;
  ip = IP(K,MaxP(n,:),P',1,0);    % 1 x NAtoms vect of inner prods
  C = C - MaxC(n) .* ip;          % update correlations
  [maxC,maxCi]=max(abs(C));       % get max abs corr. value and index
  MaxC(n+1) = C(maxCi);  MaxP(n+1,:) = P(maxCi,:);
  if(DEBUG),
    cip = MaxC(n)*ip(maxCi);
    DisplayParams('MZMP: (before opt) MaxP(n+1,:) = ',MaxP(n+1,:),4);
    DisplayParams('MZMP:              MaxC(n+1) = ',MaxC(n+1),1);
    disp(sprintf('MZMP:              MaxC(%d)*ip(%d) = (%g, %g)',...
        n,maxCi,real(cip),imag(cip)));
  end;

  %%% II.b. OPTIMIZE %%% (only for non-Dirac, modulated atoms)
  if(MaxP(n+1,1)>0 & MaxP(n+1,3)>0),  % non-Dirac atoms with pos. freq.
    j0 = MaxP(n+1,1); pj0 = pow2(j0);
    u0 = MaxP(n+1,2); f0 = MaxP(n+1,3); K0 = MaxP(n+1,4);
    fopt = f0 + (-OPT:OPT)/(pow2(OPT)*pj0);
    if(DEBUG & find(fopt > 1)), error('freq must be <= 1'); end;

    [g, Ks] = gaborMZ(K,j0,u0,fopt);  % Nopts rows, one atom per row
    Copt = x*g';                      % 1 x Nopts row of ips
    Popt = [j0*ones(Nopts,1) u0*ones(Nopts,1) fopt' K0*ones(Nopts,1)];
    ip = IP(K,MaxP(1:n,:),Popt',0,0); % n x Nopts matrix of inner prods
    Copt = Copt - MaxC(1:n)*ip;       % update correlations
    [maxCopt,maxCoptI]=max(abs(Copt));% get opt max corr value and index
    if(maxCopt > maxC),
      MaxC(n+1) = Copt(maxCoptI); MaxP(n+1,3) = fopt(maxCoptI);
      % MaxP(n+1,4)=Ks; % (no change)
      if(DEBUG),
        hNopts = (Nopts+1)/2;
        cip0 = MaxC(n)*ip(n,hNopts); cip1 = MaxC(n)*ip(n,maxCoptI);
        disp(sprintf('MZMP: (compare) |MaxC| = %g, |MaxCopt| = %g',...
            maxC,maxCopt));
        DisplayParams('MZMP: (after opt)  MaxP(n+1,:) = ',MaxP(n+1,:),4);
        DisplayParams('MZMP:              MaxC(n+1) = ',MaxC(n+1),1);
        disp(sprintf('MZMP:              MaxC(%d)*ip(%d)=(%g, %g)',...
            n,hNopts,real(cip0),imag(cip0)));
        disp(sprintf('MZMP:              MaxC(%d)*ip(%d)=(%g, %g)',...
            n,maxCoptI,real(cip1),imag(cip1)));
      end;
    end;
  end;
  %% Relative norm of estimated remainder %%
  normDiff = normRx^2 - abs(MaxC(n+1))^2;
  normRx = sqrt(abs(normDiff)); RxoxE(n+1) = normRx/normx;
  if(DEBUG),
```

```
    if(normDiff < 0),
      disp('WARNING: |correlation| > |remainder|');
      disp(sprintf('MZMP: %g = abs(MaxC(%d)) > normRx = %g',...
          abs(MaxC(n+1)),n+1,normRx));
    end;
    DisplayParams('MZMP: est. |Rx|/|x| = ',RxoxE(n+1),1);
    disp(sprintf('MZMP: iteration %d complete. hit any key...',n));pause;
  end;
  if( RxoxE(n+1) < TOLERANCE ),
    MaxC = MaxC(1:n+1);  MaxP = MaxP(1:n+1,:);  RxoxE = RxoxE(1:n+1);
    if(DEBUG),
      disp(sprintf('tol. reached at iteration %d:  |Rx|/|x| < %1.6f',...
          n+1,TOLERANCE));
    end;
    n = MAXITERS; % terminate while loop
  end;
end;

if(DEBUG==1), xEst = Atoms2Sig(K,MaxC,MaxP); end; % construct est signal
DISP=1;
if(DEBUG & (DISP > 0)),
  figure(2); clf; % plot: signal, estimated signal, and error
  subplot(3,1,1); plot(real(x)); grid; axis([0,512,-1.0,1.0]);
  subplot(3,1,2); plot(real(xEst)); grid; axis([0,512,-1.0,1.0]);
  subplot(3,1,3); plot(real(x)-real(xEst));grid; axis([0,512,-1.0,1.0]);
  figure(3); clf; % plot: signal, estimated signal, and error
  subplot(3,1,1); plot(angle(x));
  subplot(3,1,2); plot(angle(xEst));
  subplot(3,1,3); plot(angle(x)-angle(xEst));
  figure(4); clf; % norm of error at each iteration
  if(DEBUG>1),
    subplot(2,1,1);stem(real(RxoxT));
    subplot(2,1,2);stem(real(RxoxE));
  else,
    stem(real(RxoxE));
  end;
end;

%% end of file: MZMP.m
```

E.3 Wigner Transform

```
function afwig = WVTrans_AFC(sig1,sig2,DISP)
% WVTrans_AFC -- Alias-Free Cross Wigner Transform
%  Usage
%    afwig = WVDist_AFC(sig1,sig2,1)
%  Inputs
%    sig1    1-d signal
%    sig2    1-d signal
%    DISP    DISP=1 => produce image plot of Wigner Transform
%            DISP=0 => do not produce image plot (default)
```

```
%  Outputs
%    afwig   complex-valued matrix representing the alias-free
%            cross Wigner-Ville distribution of zero-extended signal
%            with rows corresponding to frequencies and columns
%            corresponding to times.
%
%  Side Effects
%    Image Plot of the alias-free Wigner-Ville distribution
%    (if DISP>0)
%
%  See Also
%    WVDist, WVDist_AF, ImagePhasePlane
%
%  References
%   Jechang Jeong and William J. Williams,
%   "Alias-Free Generalized Discrete-Time Time-Frequency Distribution,"
%   IEEE Transactions on Signal Processing, vol. 40, pp. 2757-2765.
%
if nargin < 2,
  WVDist_AF(sig1);
else,
  if nargin < 3, DISP=0; end;

  sig1 = sig1(:); sig2 = conj(sig2); sig2 = sig2(:);
  n = length(sig1);
  if(n~=length(sig2)), error('signals must have equal length'); end;
  zerosn = zeros(n,1);
  f1 = [zerosn; sig1; zerosn]; f2 = [zerosn; sig2; zerosn];
  afwig = zeros(n, n);
  ix  = 0:(n/2-1);

  for t=1:n,
    tplus  = n + t + ix;
    tminus = n + t - ix;
    x = zerosn;
    % even indices
    x(1:2:n) = f1(tplus) .* f2(tminus);
    % odd indices
    x(2:2:(n)) = (f1(tplus+1).*f2(tminus) + f1(tplus).*f2(tminus-1))/2;
    afwig(:, t) = 2* (fftshift(fft(x)));
  end

  %% display
  if(DISP),
    abstf = real(afwig);
    tfmax = max(max(abstf));
    tfmin = min(min(abstf));
    colormap(1-gray(256))
    image(linspace(0,1,n),linspace(0,1,n),256*(abstf-tfmin)/(tfmax-tfmin));
    axis('xy')
    title('Alias Free Wigner Distribution');
```

```
    xlabel('Time')
    ylabel('Frequency')
  end;
end;

%%%
% MODIFIED (to handle *cross* Wigner transform)
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 May 25
% Copyright (c) 2001, William DeMeo
%%%

% Copyright (c) 1994-5, Shaobing Chen

% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu

%% end of file: WVTrans_AFC.m
```

E.4 Gabor Atoms

```
function [g, Ks] = gaborMZ(k,s,p,f,Disp,DEBUG)
%
% Matlab code for computing complex gabor atom used in Matching
% pursuit algorithm of Mallat and Zhang
%
% Inputs
%    k    (default k=14) gabor window (atom) will have 2^(k+1)
%          sample values.  Window half-length is w = 2^k
%    s    (default s=k+1)  scale is S = 2^s
%         (S = effective support = # samples for which g > eps)
%    p    (default p=0 <--> center at w = 2^k.)
%          integer in {-2^(k+1)+1,...,-1,0,1,2,...,2^(k+1)-1}
%          equal to amount by which window is translated
%    f    (default f=0 <--> no modulation)
%          real number in {0,2,...,2^(k+1)-1}/2^(k+1)
%          specifying the *normalized* modulation frequency
%    Disp (default Disp=0 <--> do not produce plot)
%          indicator to decide whether to plot window
%    DEGUB (default DEBUG=0 <--> do not print debugging info)
%          indicator to decide whether to print debugging info
%
% Outputs
%    g    complex row vector of length 2^(k+1) containing atom values
%    Ks   the normalization constant by which g was multiplied
%
% Examples
% >> g = gabor; plot(real(g)); % default (512 sample points)
```

```
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 May 4
% Copyright (c) 2001, William DeMeo

acnt=1;      if nargin < acnt,  k = 14;   end;
acnt=acnt+1; if nargin < acnt,  s = k+1;  end;
acnt=acnt+1; if nargin < acnt,  p = 0;    end;
acnt=acnt+1; if nargin < acnt,  f = 0;    end;
%% (obsolete) acnt=acnt+1; if nargin < acnt,  phi = 0; end;
acnt=acnt+1; if nargin < acnt,  Disp = 0; end;
acnt=acnt+1; if nargin < acnt,  DEBUG = 0;end;

%%% define some constants %%%
S = pow2(s);            % scale
w = pow2(k);            % window half-length
W = pow2(k+1);          % window length (samples per window)

%%% test arguments %%%
if ((s < 0) | (s > k+1)),
 error('scale argument s ust be in {0,..., k+1}');
end
if ((p<=-W) | (p>=W)),
 error(sprintf('p == %g; should have -2^(k+1) < p < 2^(k+1)',p));
end
if ((f<0) | (f>1))
  error('frequency argument f must be in {0,2,...,2^(k+1)-1}/2^(K+1)');
end

if (s==0),   % Special Case: discrete Diracs
  if(DEBUG),disp('s==0');end;
  g = zeros(1,W);
  p = mod(p+W,W); flp = floor(p); % ensure p is a positive integer
  imp = mod(w+flp-1,W)+1;         % center of Dirac impulse
  g(imp)=1; Ks=1;
else,
  n=(-w+1:w);               % discrete time parameter
  if (s < k+1),             % s must be in {1,...,k}
    g = (2^(.25)/sqrt(S)).*exp(-pi*((n/S).^2));
    gp=g;
    if (p), % translate right by p samples, otherwise leave it equal to g
      p=mod(p+W,W); flp=floor(p); % ensure p is a positive integer
      indx = mod((1:W)+flp-1,W)+1;
      gp(indx) = g;
    end;
    %%% if(DEBUG),disp('s<k+1');disp(sprintf('norm(gp) = %g',norm(gp)));
  else,      % Special Case: Fourier basis of complex exponentials
    if(DEBUG),disp('s==k+1');end;
    gp=ones(1,W);
  end;
  Ks = 1/norm(gp);    % normalization constant
```

```
  lf = length(f);
  if(lf == 1),
    gp = Ks .* gp;                % normalize
    g = gp .* exp(i*2.0*pi*f*n); % modulate
  else,
    %   new code to allow handling multiple frequencies
    gp = Ks .* gp;      % normalize
    % OLD METHOD: (too slow)
    % f = f(:);   gp = Ks .* diag(gp);% normalize
    % g = exp(i*2.0*pi*f*n)*gp;       % modulate
    g = zeros(lf,W);
    for (mm = 1:lf),
      g(mm,:) = gp .* exp(i*2.0*pi*f(mm)*n); % modulate
    end;
  end;
end;

if Disp > 0,
  if(k<6),
    figure(1); clf; stem(real(g));
  else,
    figure(1); clf; plot(real(g));
  end;
end;

%% end of file: gaborMZ.m
```

E.5 Inner Product of Gabor Atoms

```
function ip = IP(K,P1,P2,ALL2,DEBUG)
%
% Matlab code for computing inner product of two Gabor atoms with the
% specified parameters.
%
% Inputs
%    K      2^(K+1) is the size of the entire domain
%    P1     a real N1 x 4 vector of atom parameters, where
%           P1(j,1) = s1   the scale of atom j is 2^s
%           P1(j,2) = p1   amount by which atom j is translated in time
%           P1(j,3) = f1   (normalized) modulation freq. of atom j
%           P1(j,4) = Ks1  normalization constant (unused in IP.m)
%    P2     a 4 x N2 paramter vector similar to transpose of P1.
%    ALL2   specifies whether 3rd argument, P2, represents all atom
%           parameters (used to optimize algorithm)
%    DEBUG  (default 0) DEBUG=1 ==> print extra debugging information
%
% Outputs
%    ip     a N1 x N2 matrix of inner products between atoms of P1
%           and atoms of P2.
%
% See also: IPDirac.m
```

```
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 April 13
% Copyright (c) 2001, William DeMeo

nargs=3;       if nargin < nargs, error('USAGE: IP(K,P1,P2,...)'); end;
nargs=nargs+1; if nargin < nargs, ALL2=0; end;
nargs=nargs+1; if nargin < nargs, DEBUG=0; end;

N1=size(P1,1); N2=size(P2,2); ip=zeros(N1,N2);
N = pow2(K+1);  w = pow2(K);  Nt4 = 4*N; %g2=zeros(1,N); g1=zeros(1,N);
maxexp = 11.4731;  % const. for deciding if inner prod is negligeable
                   % approx = (1/pi)*log(1/eps);

if(DEBUG),disp(sprintf('IP: computing %d inner products...',N1*N2)); end;

for n1=1:N1,
  if (P1(n1,1)==0), %%% 1st arg is a Dirac (use conjugate) %%%
    next2=1;
    if(ALL2), %%% first 2^(K+3)=4*N atoms of 2nd arg are also Diracs %%%
      ip(n1,1:Nt4) = (floor(P1(n1,2))==floor(P2(2,1:Nt4)));
      next2=Nt4+1;
    end;
    for n2=next2:N2,
      ip(n1,n2) = IPDirac(K,P1(n1,2),P2(:,n2),1);
    end;
  else, %%% 1st arg is NOT a Dirac %%%
    p1=P1(n1,2); f1=P1(n1,3); K1=P1(n1,4);
    nd = 1:N; argI=(nd'-w);
    if (P1(n1,1)==K+1), %%% 1st arg is a Fourier basis element %%%
      s1=sqrt(2);
      argR1=zeros(1,N); indxs1=1:N;
    else,
      s1=pow2(P1(n1,1));
      argR1=((mod(nd-p1+N-1,N)+1-w)/s1).^2; indxs1=find(argR1<maxexp);
    end;
    %% if(indxs1), %% only proceed if where inner prod is non-negligeable
    %% But this always holds (proof: the following line never executes)
    %% if(length(indxs1)==0),disp('no non-negligeable indices');end;
    if (ALL2),
      %%% first 2^(K+3)=4*N atoms of 2nd arg are Diracs %%%
      ip(n1,1:Nt4) = IPDirac(K,P2(2,1:Nt4),P1(n1,:),0);
      %%% atoms of 2nd arg which are neither Dirac nor complex exp %%%
      for n2=(Nt4+1:N2-Nt4),
        s2=pow2(P2(1,n2)); p2=P2(2,n2); df=f1-P2(3,n2); K2=P2(4,n2);
        argR2 = ((mod(nd-p2+N-1,N)+1-w)/s2).^2;
        % of indxs1 (support of 1st atom), find those in support of 2nd
        indxs2 = find(argR2(indxs1)<maxexp);
        indxs = indxs1(indxs2);
        if(indxs),
          %%% NB: * operator implicitly takes conjugate of 2nd arg
```

```
          %%%     WHEN THE 2ND ARG IS TRANSPOSED
          ip(n1,n2)=exp(-pi.*(argR1(indxs)+argR2(indxs)))*...
              exp(i*2.0*pi*df.*argI(indxs));
          ip(n1,n2)=ip(n1,n2)*K1*K2*sqrt(2)/sqrt(s1*s2);
        end;
      end;
      for n2=(N2-Nt4+1:N2),%%% last 4*N atoms of 2nd arg are compl exp
        df=f1-P2(3,n2); K2=P2(4,n2); argR2=zeros(1,N); s2=sqrt(2);
        ip(n1,n2)=exp(-pi.*argR1(indxs1))*exp(i*2.0*pi*df.*argI(indxs1));
        ip(n1,n2)=ip(n1,n2)*K1*K2*sqrt(2)/sqrt(s1*s2);
      end;
    else, %%% ALL2==0 => we don't know form of 2nd argument %%%
      for n2=1:N2,
        if (P2(1,n2)==0),%%% 2nd arg is Dirac (don't use conj) %%%
          ip(n1,n2) = IPDirac(K,P2(2,n2),P1(n1,:),0);
        else,  %%% 2nd argument is NOT a Dirac %%%
          p2=P2(2,n2); df=f1-P2(3,n2); K2=P2(4,n2);
          if (P2(1,n2)==K+1), %%% 2nd arg is a complex exponential %%%
            s2=sqrt(2); argR2=zeros(1,N); indxs = indxs1;
          else,
            s2=pow2(P2(1,n2));
            argR2=((mod(nd-p2+N-1,N)+1-w)/s2).^2;
            indxs2 = find(argR2(indxs1)<maxexp); indxs = indxs1(indxs2);
          end;
          if(indxs),
            ip(n1,n2)=exp(-pi.*(argR1(indxs)+argR2(indxs)))*...
                exp(i*2.0*pi*df.*argI(indxs));
            ip(n1,n2)=ip(n1,n2)*K1*K2*sqrt(2)/sqrt(s1*s2);
          end;
        end;  %~~ end test for Dirac (2nd arg)
      end;  %~~ end for n2=1:N2 loop
    end;  %~~ end test for ALL2
  end; %~~ end test for Dirac (1st arg)
end; %~~ end for n1=1:N1 loop

%% end of file: IP.m
```

E.6 Inner Product with Dirac Atoms

```
function ipd = IPDirac(K, p1, P2, CONJ)
%
% Matlab code for computing inner product of Dirac impulse(s) located
% at points specified in p1, with an atom having parameters specified
% in P2.
%
% Inputs
%   K    the signal size is 2^(K+1).
%   p1   point(s) specifying the center(s) of the Dirac impulse(s).
%   P2   parameter vector specifying the atom.
%   CONJ a boolean variable to indicate whether the Dirac is meant to
%        be first or second argument of inner product.  CONJ==1 if
```

```
%        Dirac is first argument to inner product because, in that
%        case, we take the conjugate of the atom (which is the 2nd
%        argument to the inner product in this case).
% Outputs
%   ipd  a 1 x length(p1) vector of inner products of the atom with
%        Dirac impulses having centers specified by p1.

N = pow2(K+1); w = pow2(K);
f2 = P2(3); K2 = P2(4);
flps = floor(p1);               % ensure integer valued centers
imps = mod(w+flps-1,N)+1;       % center(s) of Dirac impulse(s)

if (CONJ), konj=-1; else, konj=1; end;

if (P2(1)==K+1),  % 2nd arg is a complex exponential
  ipd = K2*exp(konj*i*2.0*pi*f2*(imps-w));
elseif (P2(1)>0), % 2nd arg is neither Dirac nor complex exponential,
  s2 = pow2(P2(1)); p2 = P2(2);
  argR2=((mod(imps-p2+N-1,N)+1-w)/s2).^2;
  ipd = zeros(1,length(p1));
  indxs = find(argR2 < 11.4731);
  ipd(indxs) = (K2*2^(.25)/sqrt(s2)).*...
       exp(-pi.*argR2(indxs)).*exp(konj*i*2.0*pi*f2*(imps(indxs)-w));
else,             % 2nd arg is also a Dirac
  ipd = (flps==floor(P2(2)));
end;

%% end of file: IPDirac.m
```

F Matlab Code

F.1 Synthesis

The following Matlab code generates Figure 1 and 2:

```
% synthAnal.m
% Matlab code for synthesis of a complex tone
% based on Sethares, "Tuning, Timbre, Spectrum, Scale"
%
% sr = sampling rate
% time = duration of wave to generate (in seconds)
% freq = frequencies of partials
% amp = amplitudes of partials

sr = 44100; time = 1; freq = [220 440 440];
amp =  [1.2 1.0 1.0]; pi2 = 2*pi; t = 0:pi2/sr:pi2*time;
wave=0*ones(length(freq)+2,length(t)); env = ones(size(t));

% Compute first 3 pure sine waves at 220, 440, 440 Hz
```

```
for i = 1:length(freq)
  wave(i,:)=wave(i,:)+amp(i)*env.*cos(freq(i)*t+pi2*rand);
end
% Compute sum of sine waves 1 and 2
wave(length(freq)+1,:)= wave(length(freq)+1,:)+wave(1,:)+wave(2,:);
% Compute sum of sine waves 1 and 3
wave(length(freq)+2,:)= wave(length(freq)+2,:)+wave(1,:)+wave(3,:);

% Plot pure sine waves 1 and 2
subplot(3,1,1); plot(t(1:2000)/pi2,wave(1,1:2000));
grid; axis([0,0.04,-1.4,1.4])
set(gca,'YTick',-1:1);set(gca,'XTickLabel',[]);
subplot(3,1,2); plot(t(1:2000)/pi2,wave(2,1:2000));
grid; axis([0,0.04,-1.4,1.4])
set(gca,'YTick',-1:1);set(gca,'XTickLabel',[]);
% Plot sum of waves 1 and 2
subplot(3,1,3); plot(t(1:2000)/pi2,wave(4,1:2000));
grid; axis([0,0.04,-2.4,2.4])
xlabel('time (in seconds)');
figure
% Plot pure sine wave 1 and 3
subplot(3,1,1); plot(t(1:2000)/pi2,wave(1,1:2000));
grid; axis([0,0.04,-1.4,1.4])
set(gca,'YTick',-1:1);set(gca,'XTickLabel',[]);
subplot(3,1,2); plot(t(1:2000)/pi2,wave(3,1:2000));
grid; axis([0,0.04,-1.4,1.4])
set(gca,'YTick',-1:1);set(gca,'XTickLabel',[]);
% Plot sum of waves 1 and 3
subplot(3,1,3); plot(t(1:2000)/pi2,wave(5,1:2000));
grid; axis([0,0.04,-2.4,2.4])
xlabel('time (in seconds)');
```

F.2 Spectrum of a Sine Wave

The following Matlab code generates Figures 6, 7, 8, and 9.

```
function [y,t]=Sinewave(CPS, SR, N)
% Sinewave.m
% Matlab code for computing a sine wave and ploting it (if PLOT=1).
%
% inputs: CPS  is the Cycles Per Second (frequency) of the sine wave in Hz.
%              This is equivalent to (1/period).
%         SR   is the Sample Rate or number of sample points per second.
%         N    is the total number of samples to generate
```

```
PLOT=0;
t=(0:(N-1))/SR;
y=sin(2*pi*CPS*t);
if PLOT==1
  figure
  disp(sprintf('Genrating plot of first 4 periods...'));
  indx=find(t < 4/CPS + 1/SR & t > 4/CPS);
  plot(t(1:indx),y(1:indx)); axis([0,t(indx),-1,1]);
end

%% end of file: Sinewave.m

% sineFFT.m
%
% Matlab code to:
%  1. generate sine wave by calling function Sinewave()
%  2. take its FFT
%  3. plot the wave and FFT. (if PLOT=1)

PLOT=1;   % 1 => draw plots;
INTEGRALPERIODS=0;  % 1 => adjust frequ to fit sample size;
CPS=220;  % frequency (Hz)
N=2^15;   % sample points
SR=44100; % sample rate
LL=-1;    % (for lower limit of FFT plot)

% get a close approximation to CPS for which the sine wave
% will have a period that is an integer multiple of SR/N.
cycs=floor(N/SR *CPS);
if INTEGRALPERIODS==1
  CPS = SR/N * cycs; % new CPS
  LL=-15;
end

[y,t]=Sinewave(CPS,SR,N);
Y=fft(y);
magspec=abs(Y);

if PLOT==1
  ppc=floor(SR/CPS);
  tper=(1:ppc)/SR; lenp=length(tper);
  tper=tper+ones(size(tper))*t(N);
  lT=[t tper];
  figure;  subplot(2,1,1);
  st=(cycs-4)/CPS; % start time of 2nd to last cycle
```

```
  indx=find(t > (st-1/SR) & t <= st);
  plot(lT(indx:N),y(indx:N)); grid;
  title('Last four periods of exactly 220 Hz sine wave');
  axis([lT(indx),lT(length(lT)),-1,1]);
  hold on
  plot(lT(N+1:length(lT)),y(1:lenp),'--');grid;
  xline=[t(N) t(N)]; yline=[-1 1];
  line(xline,yline,'LineStyle','--');
  xlabel('time (seconds)');
  subplot(2,1,2);
  f=(0:(N-1)/2)*(SR/N);
  semilogy(f(1:N/4),magspec(1:N/4));grid;
  axis([0,1000,10^(LL),10^5]);
  xlabel('frequency (Hz)');
end

%% end of file: sineFFT.m

% hammingFFT.m
%
% Matlab code to:
%  1. generate sine wave by calling function Sinewave()
%  2. multiply it by a Hamming window
%  3. take its FFT
%  4. plot the wave and FFT. (if PLOT=1)

PLOT=1;    % 1 => draw frequency plots; 2 => draw hamming window
CPS=20;   % frequency (Hz)
N=2^15;    % sample points
SR=44100;  % sample rate
LL=-1;     % (for lower limit of FFT plot)
cycs=floor(N/SR *CPS);

[y,t]=Sinewave(CPS,SR,N);
ht=(0:(N-1))/N;
ham=0.54*ones(1,length(ht)) - 0.46*cos(2*pi*ht);
yh = y.*ham;

Y=fft(yh);
magspec=abs(Y);

if PLOT==2
  figure
  subplot(3,1,1); plot(t,y);grid;
  axis([t(1),t(N),-1,1]);
```

```
  subplot(3,1,2); plot(t,ham);grid;
  axis([t(1),t(N),-1,1]);
  subplot(3,1,3); plot(t,yh);grid;
  axis([t(1),t(N),-1,1]);
  xlabel('time (seconds)');
end

if PLOT==1
  ppc=floor(SR/CPS);
  tper=(1:ppc)/SR; lenp=length(tper);
  tper=tper+ones(size(tper))*t(N);
  lT=[t tper];
  figure;  subplot(3,1,1);
  st=(cycs-20)/CPS; % start time of 10th to last cycle
  indx=find(t > (st-1/SR) & t <= st);
  subplot(2,1,1);
  plot(lT(indx:N),yh(indx:N));grid;
  title('Last ten periods of exactly 220 Hz sine wave');
  axis([lT(indx),lT(length(lT)),-1,1]);
  hold on
  plot(lT(N+1:length(lT)),yh(1:lenp),'--');
  xline=[t(N) t(N)]; yline=[-1 1];
  line(xline,yline,'LineStyle','--');
  xlabel('time (seconds)');
  f=(0:(N-1)/2)*(SR/N);
  subplot(2,1,2);
  semilogy(f(1:N/4),magspec(1:N/4));grid;
  axis([0,1000,10^(LL),10^5]);
  xlabel('frequency (Hz)');
end

%% end of file: hammingFFT.m
```

F.3 Consonance

The following Matlab code generates Figure 4:

```
% 3DissCurves.m
%
% Matlab code for generating dissonance curves using the function
% dissmeasure.m.
%
% Based on Sethares, "Tuning, Timbre, Spectrum, Scale" p.45, 301.
%
freq=[220 440 880]'; amp=1;
range=2.3; inc=0.01;
```

```
figure; hold on;
% Call function dissmeasure for each interval
for i=1:length(freq)
  diss=[0];
  for alpha = 1+inc:inc:range
    f=[freq(i,1) alpha*freq(i,1)];
    a=[amp, amp];
    d=dissmeasure(f,a);
    diss=[diss d];
  end
  plot(1:inc:range,diss)
end
grid; set(gca,'XTick',1:0.0833:2);
set(gca,'XTickLabel',{'';'';'';'';'';'fourth';'';'fifth';'';'';'';'';'octave'});
axis([1,2,0,0.9]);

%% end of file: 3DissCurves.m

function d=dissmeasure(f,amp)
%
% Matlab code for calculating instrinsic dissonance of a complex tone.
% Based on Sethares, "Tuning, Timbre, Spectrum, Scale" p.301.
% Given a set of partials in f, with amplitudes in amp, this routine
% calculates the 'sensory' dissonance among them.
%
% constants for calculation (initialization only)
if ~exist('firstpass')
  Dstar = 0.24; S1=0.0207; S2=18.96; C1=5; C2=-5;
  A1=-3.51; A2=-5.75; firstpass=1;
end
N=length(f); D=0;
[f,ind] = sort(f);
ams=amp(ind);
for i=2:N
  Fmin = f(1:N-i+1);
  S = Dstar./(S1*Fmin+S2); % equation (E.3)
  Fdif = f(i:N)-f(1:N-i+1);
  a = ams(i:N).*ams(1:N-i+1);
  Dnew = a.*(C1*exp(A1*S.*Fdif)+C2*exp(A2*S.*Fdif));
  D = D+Dnew*ones(size(Dnew))';
end
d=D;

%% end of file: dissmeasure.m
```

F.4 Energy Separation

```
function [WVE,WVI,P_DEnergy,C_DEnergy] = DEnergy(sig,DEBUG,DISP,P,C);
% DEnergy.m
% Matlab code for computing energy and interference using matching
% pursuit algorithm, MZMP, Wigner transform, WVTrans_AF, and
% cross-Wigner transform, WVTrans_AFC.
%
% Inputs
%   sig       1-dimensional signal of interest
%   DEBUG     (default 1) DEBUG=0 => do not print extra debugging info
%   DISP      (default 1) DISP=0 => do not display plot of results
%   P         optional input array of atom parameters from a prior run
%   C         optional input vector of atom correlations from a prior run
%
% Outputs
%   WVE       complex-valued matrix representing the sum of the
%             Wigner transforms of each atom in the collection of atoms
%             used to represent the signal. Rows correspond to
%             frequencies and columns corresponding to times.
%   WVI       Same as WVE except that *cross* Wigner transforms are
%             summed.
%   P_DEnergy optionally stored array of atom parameters
%   C_DEnergy optionally stored vector of atom correlations
%
% Usage
%  >> pause off; [WVE,WVI,P_DEnergy,C_DEnergy]=DEnergy;
%  >> [WVE,WVI] = DEnergy(sig,1,1,P_DEnergy,C_DEnergy);
%
% See also:  MZMP, gaborMZ, WVTrans_AF, WVTrans_AFC
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 July 20
% Copyright (c) 2001, William DeMeo

if nargin < 1,
  disp('DEnergy: no signal input... proceding with synthetic signal demo.');
  K=8; N=pow2(K+1);       % signal length
  %%% synthetic signal params (signal should really be an input) %%%%
  f1 = .1; f2 = 2*f1; f3 = 3*f1; %f1 = 440/N; f2 = 2*f1; f3 = 3*f1;
  sig=atoms(N,[1,f1,pow2(K-2),.75; 1,f2,pow2(K-2),.25; ...
    pow2(K),f2,pow2(K-2),1; 3*pow2(K-1),f3,pow2(K-2),.5]);
else,
  N=length(sig); K=nextpow2(N)-1;
end;
if nargin < 2, DEBUG=1; end;
```

```
if nargin < 3, DISP=1; end;

% Matching Pursuit parameters
MAXITERS=20;
TOLERANCE=0.05;

if nargin < 5,
  if(DEBUG),disp('DEnergy: computing new atoms with MZMP...');end;
  %%% CALL MATCHING PURSUIT FUNCTION %%%
  [P,C] = MZMP(sig,MAXITERS,TOLERANCE,DEBUG);
end;
% store a local copy of parameter/correlation book of atoms
P_DEnergy = P; C_DEnergy = C;  % (and return it if requested)

% P is at most a size MAXITERSx4 array; C is at most 1xMAXITERS
M = size(P,1);     % M = number of atoms used in signal representation
WVE = zeros(N,N);
WVI = zeros(N,N);

if(DEBUG),disp('DEnergy: computing energy & interference...');end;
for n=1:M-1,
  [g1,Ks] = gaborMZ(K,P(n,1),P(n,2),P(n,3));
  WV = WVTrans_AF(g1,0);
  WVE = WVE + (abs(C(n))^2)*WV;
  for m=n+1:M,
    [g2,Ks] = gaborMZ(K,P(m,1),P(m,2),P(m,3));
    WV = WVTrans_AFC(g1,g2,0);
    WVI = WVI + 2*real(C(n)*conj(C(m))*WV);
  end;
  if(DEBUG),disp(sprintf('DEnergy: finished atom %d of %d',n,M));end;
end;

[g1,Ks] = gaborMZ(K,P(M,1),P(M,2),P(M,3));
WV = WVTrans_AF(g1,0);
WVE = WVE + (abs(C(M))^2)*WV;

if(DISP),
  figure(1);clf;
  absE = real(WVE);
  tfmax = max(max(absE));
  tfmin = min(min(absE));
  colormap(1-gray(256));
  image(linspace(0,1,N),linspace(0,1,N),256*(absE-tfmin)/(tfmax-tfmin));
  axis('xy'); title('Wigner Energy'); xlabel('Time'); ylabel('Frequency')
```

```
  figure(2);clf;
  absI = real(WVI);
  tfmax = max(max(absI)); tfmin = min(min(absI));
  colormap(1-gray(256));
  image(linspace(0,1,N),linspace(0,1,N),256*(absI-tfmin)/(tfmax-tfmin));
  axis('xy'); title('Wigner Interference'); xlabel('Time'); ylabel('Frequency');

  figure(3);clf;
  abstf = real(WVE+WVI);
  tfmax = max(max(abstf)); tfmin = min(min(abstf));
  colormap(1-gray(256));
  image(linspace(0,1,N),linspace(0,1,N),256*(abstf-tfmin)/(tfmax-tfmin));
  axis('xy'); title('Wigner Transform'); xlabel('Time'); ylabel('Frequency')
end;

%% end of file: DEnergy.m
```

F.5 Matching Pursuit

```
function [MaxP,MaxC,P_MZMP,C_MZMP]=MZMP(sig,MAXITERS,TOLERANCE,DEBUG,P,C)
% MZMP.m
% Matlab function for Mallat/Zhang Matching Pursuit (MZMP) algorithm.
%
% Inputs
%    sig       signal (or part thereof) with which to correlate atoms;
%              signal length must be a power of 2
%    MAXITERS  (default 20) stop after MAXITERS iterations, even
%              if TOLERANCE is not reached.
%    TOLERANCE (default 0.05) stop after relative error is less
%              than TOLERANCE (i.e. |Rx|/|x| < TOLERANCE)
%    DEBUG     (default 0) DEBUG=1 ==> print extra debugging information
%    P         optionally input NAtomsx4 array of all atom parameters
%    C         optionally input 1xNAtoms array of all inner products
%
% Outputs
%    MaxP   MAXITERSx4 array of atom parameters. MaxP(i,:) are the 4
%           parameters specifying the the atoms having correlation
%           MaxC(i) with the signal remainder at step i.
%    MaxC   1xMAXITERS vector where MaxC(i) is the correlation of the
%           signal remainder with max correl atom at iteration i.
%    P_MZMP optionally stored NAtomsx4 array of all atom parameters
%    C_MZMP optionally stored 1xNAtoms array of all inner products
%
% Notes
%    The total number of atoms in the dictionary is computed as follows:
%     (K+2)*4*N = (K+2)*2^(K+3) = 2^(K+3)    Diracs
```

```
%                               + 2^(K+3)*K  Gabor atoms
%                               + 2^(K+3)    Complex exponentials
%
%    This version does not consider whether the signal is real or
%    complex and, therefore, the resulting decomposition comprises
%    complex atoms. See MZRealMP for special handling of real signals.
%
% Other Notes
%    Now that this program has been tested and seems to work, all
%    major debugging conditionals have been removed in the interest of
%    efficiency.  See MZMP_DEBUG.m
%
% See also: gaborMZ.m, IP.m, DisplayParams.m
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 May 4
% Copyright (c) 2001, William DeMeo

if nargin < 1,
  if(DEBUG),
    disp('MZMP: no signal input...');
    disp('MZMP: ...proceding with synthetic signal demo.');
  end;
  K=8; N=pow2(K+1);       % signal length
  %%% synthetic signal params (signal should really be an input) %%%%
  f1 = .1; f2 = 2*f1; f3 = 3*f1; %f1 = 440/N; f2 = 2*f1; f3 = 3*f1;
  sig=atoms(N,[1,f1,pow2(K-2),.75; 1,f2,pow2(K-2),.25; ...
    pow2(K),f2,pow2(K-2),1; 3*pow2(K-1),f3,pow2(K-2),.5]);
else,
  N=length(sig); K=nextpow2(N)-1;
end;

if nargin < 2, MAXITERS = 20; end;     % max number of iterations
if nargin < 3, TOLERANCE = 0.05; end;  % stop when |Rx|/|x| < TOLERANCE
if nargin < 4, DEBUG=0; end;

x=conj(sig(:)); x = x'; % signal is stored as ROW vector x
if(isreal(x)),
  error('use MZMPReal.m for real signals');
end;
x = hilbert(x);         % make signal analytic

pK3 = pow2(K+3);
OPT=3;                  % optimization coefficient
Nopts = pow2(OPT)-1;    % number of frequencies over which to optimize
```

```
                        % e.g. 3 => optimize over 7 frequencies
NAtoms = (K+2)*pK3;     % number of atoms in dictionary
% Ex. of (K,NAtoms) pairs: (9,45056) (10,98304) (11,212992) (12,458752)

MaxP=zeros(MAXITERS,4); % store all max parameters
MaxC=zeros(1,MAXITERS); % store all max correl values
RxoxE=zeros(1,MAXITERS);% est. norm of remainder at each iter.

DEBUG=1;  % DEBUG=1 ==> print basic tests and messages
          % DEBUG=2 ==> extra msgs. (now appears only in MZMPoptDEBUG.m)
if(DEBUG),
  disp('pause is on, hit any key...'); pause;
  disp('============================');
  disp('MZMP: STEP I. Initialization');
  disp(sprintf('MZMP: %d atoms in dictionary...',NAtoms));
end;

if nargin < 6,
  P = zeros(NAtoms,4);    %%% Parameters (P), Correlations (C) for:
  C = zeros(1,NAtoms);    %   Diracs         ji = 0,
                          %   Gabor atoms    0 < j < K+1,
                          %   Fourier basis  ji = K+1
  indx = 0;
  for j0=0:K+1, pj0 = pow2(j0);
    for k1=(1:2*pj0), fn = (k1-1)/(2*pj0);
      for u1 = (1:2*N/pj0), u=(u1-1)*pj0/2;
indx = indx + 1; % == j0*pK3 + (k1-1)*pow2(K-j0+2) + u1;
[g, Ks] = gaborMZ(K,j0,u,fn);
C(indx) = x*g';  % same: C(i)=sum(x.*conj(g)); BECAUSE PRIMED!
P(indx,:) = [j0 u fn Ks];
      end;
    end;
    if(DEBUG),disp(sprintf('MZMP:       ...finished index %d',indx));end;
  end;
else,
  disp('MZMP: ...using previously computed atoms.');
end;
P_MZMP = P; C_MZMP = C;

[maxC,maxCi]=max(abs(C));  % get max abs corr. value and index
MaxC(1) = C(maxCi);        % not the same as maxC (which is an abs value)
MaxP(1,:) = P(maxCi,:);    %[j0 u fn Ks];

if(DEBUG),
  DisplayParams('MZMP: (before opt) MaxP(1,:) = ',MaxP(1,:),4);
```

```
  DisplayParams('MZMP:              MaxC(1) = ',MaxC(1),1);
end;

%%% I.b. OPTIMIZE %%% (only for non-Dirac, modulated atoms)
if(P(maxCi,1)>0 & P(maxCi,3)>0),          % non-Dirac atoms w/ pos. freq.
  j0 = P(maxCi,1); pj0 = pow2(j0); u0 = P(maxCi,2); f0 = P(maxCi,3);
  fopt = f0 + (-OPT:OPT)/(pow2(OPT)*pj0); % 2^OPT - 1 freqs around f0
  if(DEBUG & find(fopt > 1)),
    disp('WARNING: frequency values must be no greater than 1');
  end;
  [g, Ks] = gaborMZ(K,j0,u0,fopt);   % Nopts rows, one atom per row
  Copt = x*g';                       % Nopts x 1 col. of ips
  [maxCopt,maxCoptI]=max(abs(Copt)); % get opt max corr value and index
  if(maxCopt > maxC),                % use new opt freq
    MaxC(1)=Copt(maxCoptI); MaxP(1,3)=fopt(maxCoptI); % MaxP(1,4)=Ks;
    if(DEBUG),
      disp(sprintf('MZMP: (compare) |MaxC| = %g, |MaxCopt| = %g',...
  maxC,maxCopt));
      DisplayParams('MZMP: (after opt)  MaxP(1,:) = ',MaxP(1,:),4);
      DisplayParams('MZMP:              MaxC(1) = ',MaxC(1),1);
    end;
  end;
end;

%% Relative norm of estimated remainder %%
normx = norm(x);  % normx = sqrt(x*x');
normDiff = normx^2 - abs(MaxC(1))^2;
normRx = sqrt(abs(normDiff));  RxoxE(1) = normRx/normx;
if(DEBUG),
  if(normDiff < 0),
    disp('WARNING: |correlation| > |signal|');
    disp(sprintf('MZMP: %g = abs(MaxC(1)) > normx = %g',...
abs(MaxC(1)),normx));
  end;
  DisplayParams('MZMP: est. |Rx|/|x| = ',RxoxE(1),1);
  disp('MZMP:        ...STEP I. complete.');
  disp('-----------------------------------');
  disp('MZMP: Step II. Update and Iterate');
end; n=0;
while n < MAXITERS,
  n = n+1;
  if(DEBUG), disp(sprintf('MZMP: ~~~~~~~ iteration %d ~~~~~~~',n)); end;
  ip = IP(K,MaxP(n,:),P',1,0);    % 1 x NAtoms vect of inner prods
  C = C - MaxC(n) .* ip;          % update correlations
  [maxC,maxCi]=max(abs(C));       % get max abs corr. value and index
```

```
  MaxC(n+1) = C(maxCi);  MaxP(n+1,:) = P(maxCi,:);
  if(DEBUG),
    cip = MaxC(n)*ip(maxCi);
    DisplayParams('MZMP: (before opt) MaxP(n+1,:) = ',MaxP(n+1,:),4);
    DisplayParams('MZMP:              MaxC(n+1) = ',MaxC(n+1),1);
    disp(sprintf('MZMP:              MaxC(%d)*ip(%d) = (%g, %g)',...
n,maxCi,real(cip),imag(cip)));
  end;

  %%% II.b. OPTIMIZE %%% (only for non-Dirac, modulated atoms)
  if(MaxP(n+1,1)>0 & MaxP(n+1,3)>0),  % non-Dirac atoms with pos. freq.
    j0 = MaxP(n+1,1); pj0 = pow2(j0);
    u0 = MaxP(n+1,2); f0 = MaxP(n+1,3); K0 = MaxP(n+1,4);
    fopt = f0 + (-OPT:OPT)/(pow2(OPT)*pj0);
    if(DEBUG & find(fopt > 1)), error('freq must be <= 1'); end;

    [g, Ks] = gaborMZ(K,j0,u0,fopt);  % Nopts rows, one atom per row
    Copt = x*g';                      % 1 x Nopts row of ips
    Popt = [j0*ones(Nopts,1) u0*ones(Nopts,1) fopt' K0*ones(Nopts,1)];
    ip = IP(K,MaxP(1:n,:),Popt',0,0); % n x Nopts matrix of inner prods
    Copt = Copt - MaxC(1:n)*ip;       % update correlations
    [maxCopt,maxCoptI]=max(abs(Copt));% get opt max corr value and index
    if(maxCopt > maxC),
      MaxC(n+1) = Copt(maxCoptI); MaxP(n+1,3) = fopt(maxCoptI);
      % MaxP(n+1,4)=Ks; % (no change)
      if(DEBUG),
hNopts = (Nopts+1)/2;
cip0 = MaxC(n)*ip(n,hNopts); cip1 = MaxC(n)*ip(n,maxCoptI);
disp(sprintf('MZMP: (compare) |MaxC| = %g, |MaxCopt| = %g',...
    maxC,maxCopt));
DisplayParams('MZMP: (after opt)  MaxP(n+1,:) = ',MaxP(n+1,:),4);
DisplayParams('MZMP:              MaxC(n+1) = ',MaxC(n+1),1);
disp(sprintf('MZMP:              MaxC(%d)*ip(%d)=(%g, %g)',...
    n,hNopts,real(cip0),imag(cip0)));
        disp(sprintf('MZMP:              MaxC(%d)*ip(%d)=(%g, %g)',...
    n,maxCoptI,real(cip1),imag(cip1)));
      end;
    end;
  end;
  %% Relative norm of estimated remainder %%
  normDiff = normRx^2 - abs(MaxC(n+1))^2;
  normRx = sqrt(abs(normDiff)); RxoxE(n+1) = normRx/normx;
  if(DEBUG),
    if(normDiff < 0),
      disp('WARNING: |correlation| > |remainder|');
```

```
      disp(sprintf('MZMP: %g = abs(MaxC(%d)) > normRx = %g',...
  abs(MaxC(n+1)),n+1,normRx));
    end;
    DisplayParams('MZMP: est. |Rx|/|x| = ',RxoxE(n+1),1);
    disp(sprintf('MZMP: iteration %d complete. hit any key...',n));pause;
  end;
  if( RxoxE(n+1) < TOLERANCE ),
    MaxC = MaxC(1:n+1);  MaxP = MaxP(1:n+1,:);  RxoxE = RxoxE(1:n+1);
    if(DEBUG),
      disp(sprintf('tol. reached at iteration %d:  |Rx|/|x| < %1.6f',...
  n+1,TOLERANCE));
    end;
    n = MAXITERS; % terminate while loop
  end;
end;

if(DEBUG==1), xEst = Atoms2Sig(K,MaxC,MaxP); end; % construct est signal
DISP=1;
if(DEBUG & (DISP > 0)),
  figure(2); clf; % plot: signal, estimated signal, and error
  subplot(3,1,1); plot(real(x)); grid; axis([0,512,-1.0,1.0]);
  subplot(3,1,2); plot(real(xEst)); grid; axis([0,512,-1.0,1.0]);
  subplot(3,1,3); plot(real(x)-real(xEst));grid; axis([0,512,-1.0,1.0]);
  figure(3); clf; % plot: signal, estimated signal, and error
  subplot(3,1,1); plot(angle(x));
  subplot(3,1,2); plot(angle(xEst));
  subplot(3,1,3); plot(angle(x)-angle(xEst));
  figure(4); clf; % norm of error at each iteration
  if(DEBUG>1),
    subplot(2,1,1);stem(real(RxoxT));
    subplot(2,1,2);stem(real(RxoxE));
  else,
    stem(real(RxoxE));
  end;
end;

%% end of file: MZMP.m
```

F.6 Wigner Transform

```
function afwig = WVTrans_AFC(sig1,sig2,DISP)
% WVTrans_AFC -- Alias-Free Cross Wigner Transform
%  Usage
%    afwig = WVDist_AFC(sig1,sig2,1)
%  Inputs
%    sig1    1-d signal
```

```
%    sig2    1-d signal
%    DISP    DISP=1 => produce image plot of Wigner Transform
%            DISP=0 => do not produce image plot (default)
%  Outputs
%    afwig   complex-valued matrix representing the alias-free
%            cross Wigner-Ville distribution of zero-extended signal
%            with rows corresponding to frequencies and columns
%            corresponding to times.
%
%  Side Effects
%    Image Plot of the alias-free Wigner-Ville distribution
%    (if DISP>0)
%
%  See Also
%    WVDist, WVDist_AF, ImagePhasePlane
%
%  References
%   Jechang Jeong and William J. Williams,
%   "Alias-Free Generalized Discrete-Time Time-Frequency Distribution,"
%   IEEE Transactions on Signal Processing, vol. 40, pp. 2757-2765.
%
if nargin < 2,
  WVDist_AF(sig1);
else,
  if nargin < 3, DISP=0; end;

  sig1 = sig1(:); sig2 = conj(sig2); sig2 = sig2(:);
  n = length(sig1);
  if(n~=length(sig2)), error('signals must have equal length'); end;
  zerosn = zeros(n,1);
  f1 = [zerosn; sig1; zerosn]; f2 = [zerosn; sig2; zerosn];
  afwig = zeros(n, n);
  ix  = 0:(n/2-1);

  for t=1:n,
    tplus  = n + t + ix;
    tminus = n + t - ix;
    x = zerosn;
    % even indices
    x(1:2:n) = f1(tplus) .* f2(tminus);
    % odd indices
    x(2:2:(n)) = (f1(tplus+1).*f2(tminus) + f1(tplus).*f2(tminus-1))/2;
    afwig(:, t) = 2* (fftshift(fft(x)));
  end
```

```
  %% display
  if(DISP),
    abstf = real(afwig);
    tfmax = max(max(abstf));
    tfmin = min(min(abstf));
    colormap(1-gray(256))
    image(linspace(0,1,n),linspace(0,1,n),256*(abstf-tfmin)/(tfmax-tfmin));
    axis('xy')
    title('Alias Free Wigner Distribution');
    xlabel('Time')
    ylabel('Frequency')
  end;
end;

%%%
% MODIFIED (to handle *cross* Wigner transform)
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 May 25
% Copyright (c) 2001, William DeMeo
%%%

% Copyright (c) 1994-5, Shaobing Chen

% Part of WaveLab Version 802
% Built Sunday, October 3, 1999 8:52:27 AM
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail wavelab@stat.stanford.edu

%% end of file: WVTrans_AFC.m
```

F.7 Gabor Atoms

```
function [g, Ks] = gaborMZ(k,s,p,f,Disp,DEBUG)
%
% Matlab code for computing complex gabor atom used in Matching
% pursuit algorithm of Mallat and Zhang
%
% Inputs
%    k    (default k=14) gabor window (atom) will have 2^(k+1)
%          sample values.  Window half-length is w = 2^k
%    s    (default s=k+1)  scale is S = 2^s
%         (S = effective support = # samples for which g > eps)
%    p    (default p=0 <--> center at w = 2^k.)
%          integer in {-2^(k+1)+1,...,-1,0,1,2,...,2^(k+1)-1}
%          equal to amount by which window is translated
```

```
%    f    (default f=0 <--> no modulation)
%          real number in {0,2,...,2^(k+1)-1}/2^(k+1)
%          specifying the *normalized* modulation frequency
%    Disp (default Disp=0 <--> do not produce plot)
%          indicator to decide whether to plot window
%    DEGUB (default DEBUG=0 <--> do not print debugging info)
%          indicator to decide whether to print debugging info
%
% Outputs
%    g    complex row vector of length 2^(k+1) containing atom values
%    Ks   the normalization constant by which g was multiplied
%
% Examples
% >> g = gabor; plot(real(g)); % default (512 sample points)
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 May 4
% Copyright (c) 2001, William DeMeo

acnt=1;      if nargin < acnt,  k = 14;   end;
acnt=acnt+1; if nargin < acnt,  s = k+1;  end;
acnt=acnt+1; if nargin < acnt,  p = 0;    end;
acnt=acnt+1; if nargin < acnt,  f = 0;    end;
%% (obsolete) acnt=acnt+1; if nargin < acnt,  phi = 0; end;
acnt=acnt+1; if nargin < acnt,  Disp = 0; end;
acnt=acnt+1; if nargin < acnt,  DEBUG = 0;end;

%%% define some constants %%%
S = pow2(s);            % scale
w = pow2(k);            % window half-length
W = pow2(k+1);          % window length (samples per window)

%%% test arguments %%%
if ((s < 0) | (s > k+1)),
 error('scale argument s ust be in {0,..., k+1}');
end
if ((p<=-W) | (p>=W)),
 error(sprintf('p == %g; should have -2^(k+1) < p < 2^(k+1)',p));
end
if ((f<0) | (f>1))
  error('frequency argument f must be in {0,2,...,2^(k+1)-1}/2^(K+1)');
end

if (s==0),   % Special Case: discrete Diracs
  if(DEBUG),disp('s==0');end;
```

```
  g = zeros(1,W);
  p = mod(p+W,W); flp = floor(p); % ensure p is a positive integer
  imp = mod(w+flp-1,W)+1;         % center of Dirac impulse
  g(imp)=1; Ks=1;
else,
  n=(-w+1:w);                % discrete time parameter
  if (s < k+1),              % s must be in {1,...,k}
    g = (2^(.25)/sqrt(S)).*exp(-pi*((n/S).^2));
    gp=g;
    if (p), % translate right by p samples, otherwise leave it equal to g
      p=mod(p+W,W); flp=floor(p); % ensure p is a positive integer
      indx = mod((1:W)+flp-1,W)+1;
      gp(indx) = g;
    end;
    %%% if(DEBUG),disp('s<k+1');disp(sprintf('norm(gp) = %g',norm(gp)));
  else,      % Special Case: Fourier basis of complex exponentials
    if(DEBUG),disp('s==k+1');end;
    gp=ones(1,W);
  end;
  Ks = 1/norm(gp);    % normalization constant
  lf = length(f);
  if(lf == 1),
    gp = Ks .* gp;               % normalize
    g = gp .* exp(i*2.0*pi*f*n); % modulate
  else,
    %   new code to allow handling multiple frequencies
    gp = Ks .* gp;      % normalize
    % OLD METHOD: (too slow)
    % f = f(:);   gp = Ks .* diag(gp);% normalize
    % g = exp(i*2.0*pi*f*n)*gp;       % modulate
    g = zeros(lf,W);
    for (mm = 1:lf),
      g(mm,:) = gp .* exp(i*2.0*pi*f(mm)*n); % modulate
    end;
  end;
end;

if Disp > 0,
  if(k<6),
    figure(1); clf; stem(real(g));
  else,
    figure(1); clf; plot(real(g));
  end;
end;
```

```
%% end of file: gaborMZ.m
```

F.8 Inner Product of Gabor Atoms

```
function ip = IP(K,P1,P2,ALL2,DEBUG)
%
% Matlab code for computing inner product of two Gabor atoms with the
% specified parameters.
%
% Inputs
%    K      2^(K+1) is the size of the entire domain
%    P1     a real N1 x 4 vector of atom parameters, where
%           P1(j,1) = s1   the scale of atom j is 2^s
%           P1(j,2) = p1   amount by which atom j is translated in time
%           P1(j,3) = f1   (normalized) modulation freq. of atom j
%           P1(j,4) = Ks1  normalization constant (unused in IP.m)
%    P2     a 4 x N2 paramter vector similar to transpose of P1.
%    ALL2   specifies whether 3rd argument, P2, represents all atom
%           parameters (used to optimize algorithm)
%    DEBUG  (default 0) DEBUG=1 ==> print extra debugging information
%
% Outputs
%    ip     a N1 x N2 matrix of inner products between atoms of P1
%           and atoms of P2.
%
% See also: IPDirac.m
%
% Author: William DeMeo <william.demeo@verizon.net>
% Date: 2001 April 13
% Copyright (c) 2001, William DeMeo

nargs=3;       if nargin < nargs, error('USAGE: IP(K,P1,P2,...)'); end;
nargs=nargs+1; if nargin < nargs, ALL2=0; end;
nargs=nargs+1; if nargin < nargs, DEBUG=0; end;

N1=size(P1,1); N2=size(P2,2); ip=zeros(N1,N2);
N = pow2(K+1);  w = pow2(K);  Nt4 = 4*N; %g2=zeros(1,N); g1=zeros(1,N);
maxexp = 11.4731;  % const. for deciding if inner prod is negligeable
                   % approx = (1/pi)*log(1/eps);

if(DEBUG),disp(sprintf('IP: computing %d inner products...',N1*N2)); end;

for n1=1:N1,
  if (P1(n1,1)==0), %%% 1st arg is a Dirac (use conjugate) %%%
    next2=1;
    if(ALL2), %%% first 2^(K+3)=4*N atoms of 2nd arg are also Diracs %%%
```

```
      ip(n1,1:Nt4) = (floor(P1(n1,2))==floor(P2(2,1:Nt4)));
      next2=Nt4+1;
    end;
    for n2=next2:N2,
      ip(n1,n2) = IPDirac(K,P1(n1,2),P2(:,n2),1);
    end;
  else, %%% 1st arg is NOT a Dirac %%%
    p1=P1(n1,2); f1=P1(n1,3); K1=P1(n1,4);
    nd = 1:N; argI=(nd'-w);
    if (P1(n1,1)==K+1), %%% 1st arg is a Fourier basis element %%%
      s1=sqrt(2);
      argR1=zeros(1,N); indxs1=1:N;
    else,
      s1=pow2(P1(n1,1));
      argR1=((mod(nd-p1+N-1,N)+1-w)/s1).^2; indxs1=find(argR1<maxexp);
    end;
    %% if(indxs1), %% only proceed if where inner prod is non-negligeable
    %% But this always holds (proof: the following line never executes)
    %% if(length(indxs1)==0),disp('no non-negligeable indices');end;
    if (ALL2),
      %%% first 2^(K+3)=4*N atoms of 2nd arg are Diracs %%%
      ip(n1,1:Nt4) = IPDirac(K,P2(2,1:Nt4),P1(n1,:),0);
      %%% atoms of 2nd arg which are neither Dirac nor complex exp %%%
      for n2=(Nt4+1:N2-Nt4),
s2=pow2(P2(1,n2)); p2=P2(2,n2); df=f1-P2(3,n2); K2=P2(4,n2);
argR2 = ((mod(nd-p2+N-1,N)+1-w)/s2).^2;
% of indxs1 (support of 1st atom), find those in support of 2nd
indxs2 = find(argR2(indxs1)<maxexp);
indxs = indxs1(indxs2);
if(indxs),
  %%% NB: * operator implicitly takes conjugate of 2nd arg
  %%%     WHEN THE 2ND ARG IS TRANSPOSED
  ip(n1,n2)=exp(-pi.*(argR1(indxs)+argR2(indxs)))*...
      exp(i*2.0*pi*df.*argI(indxs));
  ip(n1,n2)=ip(n1,n2)*K1*K2*sqrt(2)/sqrt(s1*s2);
end;
      end;
      for n2=(N2-Nt4+1:N2),%%% last 4*N atoms of 2nd arg are compl exp
df=f1-P2(3,n2); K2=P2(4,n2); argR2=zeros(1,N); s2=sqrt(2);
ip(n1,n2)=exp(-pi.*argR1(indxs1))*exp(i*2.0*pi*df.*argI(indxs1));
ip(n1,n2)=ip(n1,n2)*K1*K2*sqrt(2)/sqrt(s1*s2);
      end;
    else, %%% ALL2==0 => we don't know form of 2nd argument %%%
      for n2=1:N2,
if (P2(1,n2)==0),%%% 2nd arg is Dirac (don't use conj) %%%
```

```
  ip(n1,n2) = IPDirac(K,P2(2,n2),P1(n1,:),0);
else,  %%% 2nd argument is NOT a Dirac %%%
  p2=P2(2,n2); df=f1-P2(3,n2); K2=P2(4,n2);
  if (P2(1,n2)==K+1), %%% 2nd arg is a complex exponential %%%
    s2=sqrt(2); argR2=zeros(1,N); indxs = indxs1;
  else,
    s2=pow2(P2(1,n2));
    argR2=((mod(nd-p2+N-1,N)+1-w)/s2).^2;
    indxs2 = find(argR2(indxs1)<maxexp); indxs = indxs1(indxs2);
  end;
  if(indxs),
    ip(n1,n2)=exp(-pi.*(argR1(indxs)+argR2(indxs)))*...
exp(i*2.0*pi*df.*argI(indxs));
    ip(n1,n2)=ip(n1,n2)*K1*K2*sqrt(2)/sqrt(s1*s2);
  end;
end;  %~~ end test for Dirac (2nd arg)
      end;  %~~ end for n2=1:N2 loop
    end;  %~~ end test for ALL2
  end; %~~ end test for Dirac (1st arg)
end; %~~ end for n1=1:N1 loop

%% end of file: IP.m
```

F.9 Inner Product with Dirac Atoms

```
function ipd = IPDirac(K, p1, P2, CONJ)
%
% Matlab code for computing inner product of Dirac impulse(s) located
% at points specified in p1, with an atom having parameters specified
% in P2.
%
% Inputs
%   K    the signal size is 2^(K+1).
%   p1   point(s) specifying the center(s) of the Dirac impulse(s).
%   P2   parameter vector specifying the atom.
%   CONJ a boolean variable to indicate whether the Dirac is meant to
%        be first or second argument of inner product.  CONJ==1 if
%        Dirac is first argument to inner product because, in that
%        case, we take the conjugate of the atom (which is the 2nd
%        argument to the inner product in this case).
% Outputs
%   ipd  a 1 x length(p1) vector of inner products of the atom with
%        Dirac impulses having centers specified by p1.

N = pow2(K+1); w = pow2(K);
f2 = P2(3); K2 = P2(4);
```

```
flps = floor(p1);               % ensure integer valued centers
imps = mod(w+flps-1,N)+1;       % center(s) of Dirac impulse(s)

if (CONJ), konj=-1; else, konj=1; end;

if (P2(1)==K+1),  % 2nd arg is a complex exponential
  ipd = K2*exp(konj*i*2.0*pi*f2*(imps-w));
elseif (P2(1)>0), % 2nd arg is neither Dirac nor complex exponential,
  s2 = pow2(P2(1)); p2 = P2(2);
  argR2=((mod(imps-p2+N-1,N)+1-w)/s2).^2;
  ipd = zeros(1,length(p1));
  indxs = find(argR2 < 11.4731);
  ipd(indxs) = (K2*2^(.25)/sqrt(s2)).*...
       exp(-pi.*argR2(indxs)).*exp(konj*i*2.0*pi*f2*(imps(indxs)-w));
else,             % 2nd arg is also a Dirac
  ipd = (flps==floor(P2(2)));
end;

%% end of file: IPDirac.m
```

References

[1] Sergio Cavaliere and Aldo Piccialli, *Granular synthesis of musical signals*, pp. 155–186, In Roads et al. [2], 1997.

[2] Curtis Roads, Stephen Travis Pope, Aldo Piccialli, and Giovanni De Poli, Eds., *Musical Signal Processing*, Swets & Zeitlinger, Lisse, The Netherlands, 1997.

[3] H. Helmholtz, *On the Sensations of Tone*, Dover, New York, 1877.

[4] R. Plomp and W. J. M. Levelt, "Tonal consonance and critical bandwidth," *Journal of the Acoustical Society of America*, vol. 38, pp. 548–560, 1965.

[5] William A. Sethares, *Tuning, Timbre, Spectrum, Scale*, Springer-Verlag, 1997.

[6] William M. Hartmann, *Signals, Sound, and Sensation*, Springer-Verlag, 1998.

[7] X. Rodet and P. Depalle, "A new additive synthesis method using inverse fourier transform and spectral envelopes," in *Proceedings of the International Compter Music Conference (ICMC '92)*, San Jose, October 1992.

[8] J. Tenney, *A History of 'Consonance' and 'Dissonance'*, Excelsior Music, 1988.

[9] D. Gabor, "Theory of communication," *Journal of the IEE*, vol. 93, pp. 429–457, 1946.

[10] Stéphane Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.

[11] N. Delprat, B. Escudié, P. Guillemain, R. Kronland-Martinet, P. Tchamitchian, and B. Torrésani, "Asymptotic wavelet and gabor analysis: extraction of instantaneous frequencies," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 644–664, March 1992.

[12] S.G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Proc.*, vol. 41, pp. 3397–3415, 1993.

[13] Patrick Flandrin, *Time-Frequency/Time-Scale Analysis*, Academic Press, 1999.

[14] R. Gribonval, P. Depalle, X. Rodet, E. Bacry, and S. Mallat, "Sound signal decomposition using a high resolution matching pursuit," in *Proceedings of the International Computer Music Conference*. International Computer Music Association, 1996.

[15] Anthony Teolis, *Computational Signal Processing with Wavelets*, Birkhäuser, 1998.

[16] J. E. Moyal, "Quantum mechanics as a statistical theory," in *Proc. Cambridge Phi. Soci.*, 1949, vol. 45, pp. 99–124.

[17] Joseph Kermin and Gary Tomlinson, *Listen*, Bedford/St. Martin's, brief fourth edition, 2000.