A MULTIPITCH ESTIMATION ALGORITHM BASED ON FUNDAMENTAL FREQUENCIES AND PRIME HARMONICS

Arturo Camacho
School of Computer Science and Informatics
University of Costa Rica
arturo.camacho@ecci.ucr.ac.cr

Iosef Kaver-Oreamuno
Research Center for Info. & Comm. Techs.
University of Costa Rica
iosefkaver@gmail.com

ABSTRACT

An algorithm named Prime-multiF0 for the estimation of multiple pitches in a signal is proposed. Unlike other algorithms that consider all harmonics of every pitch candidate, our algorithm considers only on the fundamental frequency and prime harmonics. This approach is shown to work extremely well with chords made of intervals no smaller than a minor third. A test suite was created using synthetic signals of sawtooth, square, and triangle waves; major, minor, diminished and augmented triads in fundamental and first and second inversion, and spanning a bass range of three octaves. Experimental results show that our algorithm was able to detect the correct notes (after rounding to the closest semitone) for all the sawtooth and square waves in the test set, and for 99.3% of the triangle waves, failing only on very high pitch notes. An implementation of the algorithm in MATLAB is provided.

1. INTRODUCTION

The first attempts to solve the problem of estimating pitch in monophonic signals can be traced back to the 1960s, according to a review of early algorithms in [1]. More recent and successful approaches to solve the problem are presented in [2] and [3]. These algorithms can be applied to solve problems in speech coding [4], speech therapy [5], and music information retrieval [6], but they fail to solve complex music problems like transcription, which require estimating the pitch of concurrent signals [7].

In solving the multipitch problem, [8] and [9] are among the most successful algorithms. They both use an auditory model to split the signal in bands (notably two in [8]), apply half-wave rectification to generate extra harmonics, compute a generalized summary autocorrelation function (SACF), and process this SACF to obtain the different pitches from its peaks. In [8] the process consists in enhancing the SACF by setting negative values to zero and subtracting from the SACF stretched copies of itself to eliminate spurious peaks (notably the root of the chord). In [9] the most salient peak of the SACF is used to estimate the pitch of one of the signals. Then, an attempt is made to remove the recognized signal from the spectrum, and the SACF is recomputed on the modified spectrum to recognize another pitch. The process is repeated for the other sound sources.

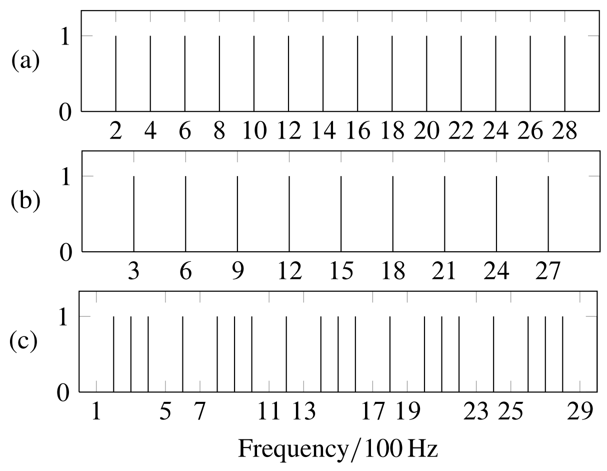

Figure 1: Spectral components of (a) a signal with fundamental frequency 200 Hz, (b) a signal with fundamental frequency 300 Hz, and (c) a signal combining both 200 Hz and 300 Hz signals. In the latter case, the numbers along the horizontal axis are the frequencies for which there are no harmonics in the spectrum, which would correspond to missing harmonics of a signal whose fundamental frequency is 100 Hz (the maximum common divisor of 200 Hz and 300 Hz).

The approaches in [8] and [9] use a generalized autocorrelation function consisting in the inverse Fourier transform of the magnitude of the signal's modified spectrum (with extra harmonics), raised to a certain power. A closer look reveals that this function computes a score for each pitch candidate based on the number and magnitude of the peaks found at harmonics of the candidate in the modified spectrum. To illustrate the rationale behind this idea, Fig. 1(a) and (b) shows the spectral components of two harmonic signals with fundamental frequencies 200 Hz and 300 Hz, and Fig. 1(c) shows the components of the combined signal. The signals are assumed to be limited to a maximum frequency of 3 kHz (they are sampled at 6 kHz), which means that the 200 Hz tone has 15 harmonics and the 300 Hz tone has 10 harmonics. Based on the scoring schema stated above, their respective scores would be 15 and 10. However, since 100 Hz is the root of the chord (i.e., all harmonics are multiples of 100), that frequency would receive a score of 20, corresponding to the total number of harmonics in (c). The score for the (combined) 100 Hz signal would then be higher than the score for the individual tones, which would cause autocorrelation to fail at estimating the correct fundamental frequency of these types of combined signals.

In order to overcome this shortcoming of autocorrelation, a different approach could be used that gives credit to a pitch candidate based solely on those peaks located at its fundamental frequency and prime harmonics. Using this approach, the 200 Hz candidate would receive a score of 7 (for the components at 2, 4, 6, 10, 14, 22, and 26 in Fig. 1-a), the 300 Hz candidate would receive a score of 5 (for the components at 3, 6, 9, 15, and 21 in Fig. 1-b), and the 100 Hz candidate would receive score of 2 (for the components at 2 and 3 in Fig. 1-c), leaving 200 Hz and 300 Hz as the indisputable winners. It can be shown that no candidate can receive credit for more than one harmonic of any other tone. This approach was successfully used in [3] for the single pitch estimation problem, and will be used in the algorithm presented here, [1] together with a variation of the enhancement proposed in [8], for multiple pitch detection.

The rest of the paper is organized as follows. Section 2 describes the proposed method, Section 3 presents and discusses the results, Section 4 outlines our conclusions, and the Appendix contains the implementation of the proposed algorithm in MATLAB.

2. METHOD

The proposed method, named Prime-multiF0, works as follows. It divides the signal in windows, computes the spectrum in each window, and applies an integral transform to the spectrum to obtain a score for each pitch candidate. Then, negatives scores are set to zero and a subharmonic subtraction step is performed to enhance the scores. Finally, a peak-selection method is applied to detect the different pitches. The details of the method are described next.

To compute the score of a pitch candidate f, the authors of [3] recommend to use of a Hann window of size $8/f$. The use of this window type and size makes the width of the spectral lobes match the width of the positive part of of the kernel of the transform. This means that the spectrum of the signal at time t should be computed as:

$$X(t,f') = \int_{-\infty}^{\infty} w\left(8/f, t'-t\right) x\left(t'\right) e^{-2\pi i f' t'} dt', \quad (1)$$

where

$$w(T,t) = \begin{cases} \frac{1}{2}\left(1+\cos\left(\pi(t'-t)/T\right)\right), & \text{if } |t| < T/2 \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

[1] Surprisingly, some of the other novel features proposed in [3] (the use of the psychoacoustics based ERB scale, the weighting of the harmonics inversely proportional to frequency, and the normalization of the kernel) did not work well in this new context and were thus removed.

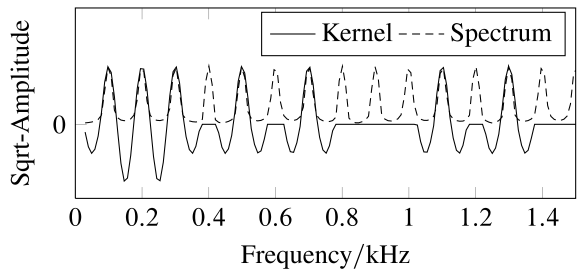

Figure 2: Spectrum of a signal with fundamental frequency 100 Hz and kernel that allows to recognize its pitch.

Then, the score of a candidate f at time t should be computed as the integral transform

$$S(t,f) = \int_0^{\infty} K\left(f,f'\right) |X(t,f)|^{1/2}\, df' \quad (3)$$

with kernel

$$K(f,f') = K_1(f,f') + \sum_{j\in\mathbb{P}} K_j(f,f'), \quad (4)$$

where

$$K_j(f,f') = \begin{cases} \cos(2\pi f'/f), & \text{if } |f'/f - j| < \frac{1}{4} \\ \frac{1}{2}\cos(2\pi f'/f), & \text{if } \frac{1}{4} < |f'/f - j| < \frac{3}{4} \\ 0, & \text{otherwise,} \end{cases} \quad (5)$$

and $\mathbb{P}$ is the set of prime numbers. Each component of the kernel has the purpose to give credit to the candidate if there is energy at its j-th harmonic. [2] This is illustrated in Fig. 2, which shows the spectrum of a signal with fundamental frequency 100 Hz and the kernel with same frequency.

Unfortunately, it is computationally expensive to compute a different Fourier transform for each pitch candidate. Hence, we adopt a schema to reduce the number of transforms computed, at the price of a decreased match between the width of the spectral lobes and the width of the components of the kernel. Under this scheme, transforms are computed using only window sizes that are powers of two (in number of samples), and the score of each candidate is produced as a linear combination of the scores obtained for that candidate using the two closest power-of-two window sizes. More precisely, the scores are computed as follows:

$$S(t,f) = (1-\lambda)\, S_0(t,f) + \lambda S_1(t,f) \quad (6)$$

where $S_0(t,f)$ and $S_1(t,f)$ are computed as in (3), but using the two closest power-of-two window sizes $N_0 = 2^L$ and $N_1 = 2^{L+1}$ (in samples), respectively; and λ is obtained from N^*, the optimal window size (in samples) for a candidate f, with $N^* = 2^{L+\lambda} = [8f_s/f]$, where

[2] The use of the square-root amplitude of the spectrum in (3) approximates the growth of loudness with amplitude [10], and the use of a cosine in (5) allows for inharmonicity in the signals. Both are explained in more detail in [3].

$L \in \mathbb{N}, 0 \leq \lambda < 1$, and f_s is the sampling rate of the signal).

Finally, the scores computed in (6) are enhanced by applying a variation of the approach proposed in [8]. The original enhancement consists in setting to zero all negative scores and subtracting from each score the scores of multiples of the candidate. This has the purpose of reducing the scores of the inevitable peaks at common divisors of the pitches (e.g., the root of the chord). The proposed variation uses only scalings by a prime factor:

$$S'(t,f) = S^+(t,f) - \sum_{k \in \mathbb{P}} S^+(t,kf), \tag{7}$$

where

$$S^+(t,f) = \max\{0, S(t,f)\} \tag{8}$$

is a clipped version of $S(t,f)$ to nonnegative values.

Ideally, one would wish to use (7) itself to determine the pitch of the notes being played at time t, but $S'(t,f)$ is a little bit unstable over short periods of time. Therefore, it is recommended to integrate it over some period of time in order to obtain a more reliable estimate of the notes' pitch. In our experiments we obtained good results integrating over 0.3 s.

The algorithm implementation that we used is provided in Sec. 5 (Appendix). It is written in the MATLAB programming language. The integrals in (1) and (3) were approximated by sums using as step sizes $\Delta t' = 1/f_s$ and $\Delta f' = f_s/N$, where f_s is the sampling frequency of the signal and N is the window size (in samples).

3. RESULTS

In order to test the algorithm, we used chords consisting of major, minor, diminished and augmented triads. All triads were played in root position, first inversion and second inversion, except for augmented chords, for which inversions are indistinguishable from other augmented chord in root position. This made for a total of ten chord profiles. The bass was let run from C_3 (about 130.8 Hz) to B_5 (about 987.8 Hz) for a total of 36 different basses and 360 chords. The chords were generated using synthetic signals sampled at a rate of 48 kHz and with a duration of 0.3 s. The signals consisted of sawtooth, square, and triangle waves. These types of signals were chosen because of their popularity in the literature and their interesting spectral characteristics:

Sawtooth waves The amplitude of their harmonics decays inversely proportional to frequency.

Square waves Have only even harmonics. Their amplitude decays inversely proportional to frequency.

Triangle waves Have only even harmonics. Their amplitude decays inversely proportional to the square of frequency.

Each chord was built using only signals of the same type and each type was used to build each of the chords. This made for a total of 1440 chords. Since every chord has three notes, the total number of notes was 4320.

The performance of the algorithm was compared to that one in [9]. The pitch search range used to test the algorithms was from 30 to 5 kHz (based on psychoacoustic experiments [11–13]) and the resolution was a quarter of a semitone. For the proposed algorithm, the candidate with the highest score was matched to the closest note of the chord, the candidate with the second highest score was matched to the next available closest note (as long as the distance to the previously detected note exceeds 2.5 semitones), and the third best candidate was matched to the last note of the chord (as long as the distance to the previously detected notes exceeds 2.5 semitones). [3] Since the notes of the chords and the pitch candidates shared the same tuning ($A_4 = 440$ Hz) and the resolution used for the candidates was one quarter of a semitone, all errors were a multiple of that quantity. For Klapuri's algorithm, we matched each note outputted by the algorithm to the closest note of the chord. A note was considered to be in error if its distance to the assigned note exceeded half a semitone.

Table 1: Error rates for the evaluated algorithms on sawtooth, square, and triangle waves.

	Error rate (%)	
Signal type	Prime-multiF0	Klapuri
Sawtooth waves	0.00	0.00
Square waves	0.00	0.74
Triangle waves	0.74	0.28

Table 1 shows the error rates classified by signal type. The proposed algorithm (Prime-multiF0) produced no errors for sawtooth and square waves, and an error rate of 0.74% for triangle waves. On the other hand, the algorithm proposed by Klapuri produced no errors for sawtooth waveforms, but error rates of 0.74% and 0.28% for square and triangle waves, respectively.

Errors tend to occur only at the very high and low ends of the chosen pitch range for the signals. This is illustrated in Fig. 3, which shows the cumulative error rate as a function of bass. Errors from Prime-multiF0 occur only with basses above E_5 (660 Hz), and errors from Klapuri's algorithm occur in a similar range for square waves, and with basses below G_3 (196 Hz) for triangle waves. This means that both algorithms produce good results for the most common pitch range for chords (i.e., the *middle* octave: between C_4 and B_4).

For illustrative purposes, we show in Figs. 4–6 the average scores produced by Prime-multiF0 for each type of chord as a function of the distance to the bass in semitones. [4]

[3] Candidates located at less than 2.5 semitones of previously recognized pitches are not considered because we empirically found that the algorithm, in its current state, does not work well for intervals smaller than a minor third (three semitones).

[4] The scores were normalized to the maximum score in each chord before taking averages. This was done to avoid a bias toward chords with overall higher scores, and to reduce the variance in Fig. 4. Averages were taken over different signal types (i.e., sawtooth, square, and triangle waves.)

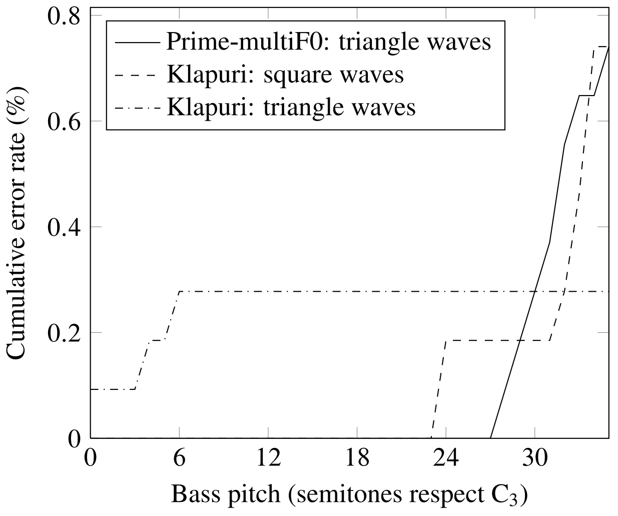

Figure 3: Cumulative error rates as a function of distance of the bass respect C_3. Most errors occur at the low or high ends of the bass range used in the experiment.

4. CONCLUSION

A new estimation algorithm for multipitch signals was proposed. This algorithm, named Prime-multiF0, was successful in detecting the pitch of chords played with synthetic signals consisting of sawtooth, square and triangle waves, for a wide range of pitch. Its performance was slightly better than a well known algorithm in the literature. However, both algorithms performed exceedingly well, and more tests with natural signals need to be performed to obtain more realistic results.

5. APPENDIX: IMPLEMENTATION IN MATLAB

```
function S = primemultif0(x,fs,pc,dt)
% S = PRIMEMULTIF0(X,Fs,PC,DT) computes
% pitch scores for the pitch candidates PC
% every DT seconds, based on the signal X
% with sampling frequency FS. PC must be an
% increasing sequence.
t = [0:dt:length(x)/fs]'; % time
pc = pc(:);
log2pc = log2(pc);
S = zeros(length(pc),length(t)); % scores
logWs = round(log2(8*fs./pc));
ws = 2.^[logWs(1):-1:logWs(end)]; % window
    sizes
pO = 8*fs./ws; % optimal pitches for window
    sizes
% Determine window sizes used by each pitch
    candidate:
d = 1 + log2pc - log2(8*fs./ws(1));
for i=1:length(ws)
    dn = max(1,round(4*fs/pO(i))); % hop
        size
    % Zero-pad signal:
    xzp = [zeros(ws(i)/2,1);x(:);zeros(dn+
        ws(i)/2,1)];
    % Compute spectrum:
    w = hanning(ws(i)); % Hann window
    o = max(0,round(ws(i) - dn)); % window
        overlap
```

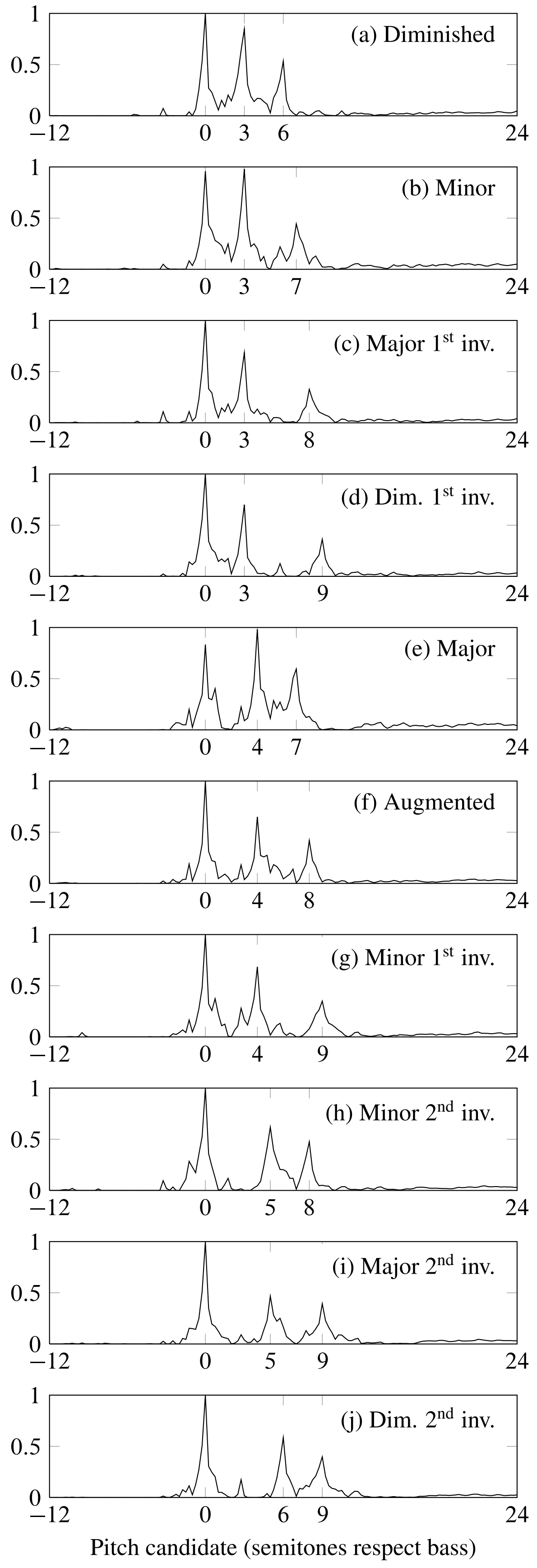

Figure 4: Average scores produced by the Prime-multiF0 algorithm for chords produced using *sawtooth* waveforms.

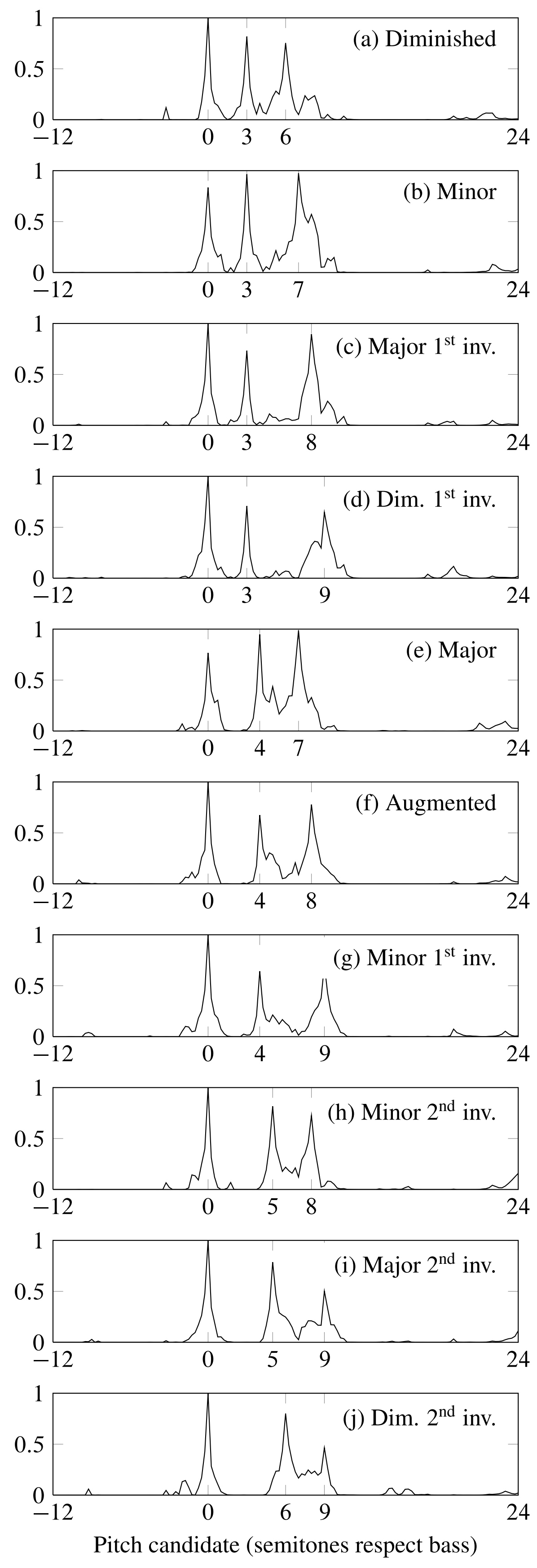

Figure 5: Average scores produced by the Prime-multiF0 algorithm for chords produced using *square* waveforms.

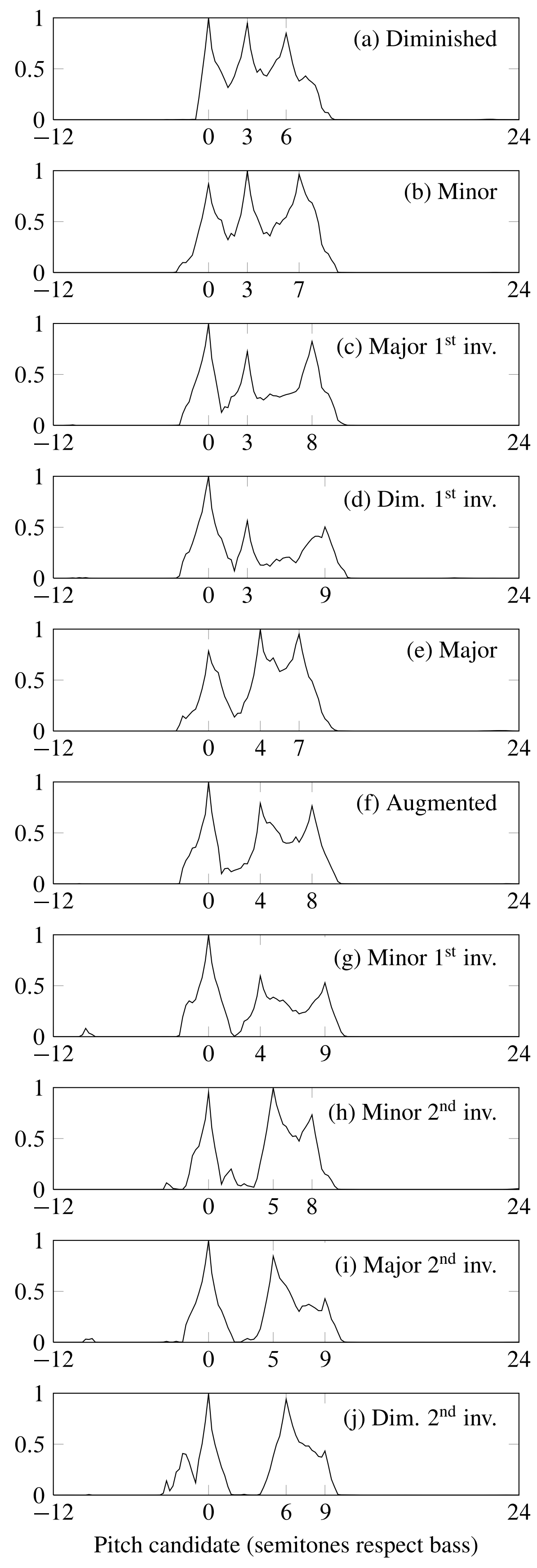

Figure 6: Average scores produced by the Prime-multiF0 algorithm for chords produced using *triangle* waveforms.

```
    [X,f,ti] = specgram(xzp,ws(i),fs,w,o);
    % Select candidates that use this
        window size:
    if length(ws) == 1
        j=(1:length(pc))'; k=[];
    elseif i == length(ws)
        j=find(d-i>-1); k=find(d(j)-i<0);
    elseif i==1
        j=find(d-i<1); k=find(d(j)-i>0);
    else
        j=find(abs(d-i)<1); k=1:length(j);
    end
    % Compute loudness at required
        frequency range:
    first = find(f>pc(j(1))/4,1,'first');
    f = f(first:end);
    L = sqrt(abs(X(first:end,:)));
    % Compute scores:
    Si = scoresAllCandidates(f,L,pc(j));
    Si = interp1(ti,Si',t,'linear',NaN)';
    lambda = d(j(k)) - i;
    mu = ones(size(j));
    mu(k) = 1 - abs(lambda);
    S(j,:) = S(j,:) + repmat(mu,1,size(Si
        ,2)).*Si;
end
% Enhance pitch strength matrix:
S = max(0,S);
for i = primes(pc(end)/pc(1));
    S = S - max(0,interp1(pc,S,i*pc,'linear
        ',0));
end
S = max(0,S);

function S = scoresAllCandidates(f,L,pc)
% Create score matrix:
S = zeros(length(pc),size(L,2));
% Define integration regions:
k = ones(1,length(pc)+1);
for j = 1:length(k)-1
    k(j+1) = k(j) - 1 + find(f(k(j):end)>pc
        (j)/4,1,'first');
end
k = k(2:end);
% Create loudness normalization matrix:
N = (flipud(cumsum(flipud(L))));
for j = 1:length(pc)
    % Normalize loudness:
    n = N(k(j),:);
    n(n==0) = Inf; % to make zero-loudness
        equal zero after normalization
    NL = L(k(j):end,:)./repmat(n,size(L,1)-
        k(j)+1,1);
    % Compute score:
    S(j,:) = scoreOneCandidate(f(k(j):end),
        NL,pc(j));
end

function S = scoreOneCandidate(f,NL,pc)
n = fix(f(end)/pc-0.75); % number of
    harmonics
if n==0, S=NaN; return, end
k = zeros(size(f)); % kernel
q = f/pc; % normalized frequency w.r.t.
    candidate
% Create kernel:
for i = [1 primes(n)]
    a = abs(q-i);
    % Peak's weigth:
    p = a<0.25;
    k(p) = cos(2*pi*q(p));
    % Valleys' weights
    v = 0.25<a & a<0.75;
    k(v) = k(v) + cos(2*pi*q(v))/2;
end
S = k'*NL;
```

6. REFERENCES

[1] W. Hess, *Pitch Determination of Speech Signals*. Springer, 1983.

[2] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 111, no. 4, pp. 1917–1930, 2002.

[3] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 124, pp. 1638–1652, 2008.

[4] A. S. Spanias, "Speech coding: a tutorial review," *Proceedings of the IEEE*, vol. 82, no. 10, pp. 1541 –1582, 1994.

[5] J. Hillenbrand and R. A. Houde, "Acoustic correlates of breathy vocal quality: Dysphonic voices and continuous speech," *Journal of Speech, Language and Hearing Research*, vol. 39, no. 2, p. 311, 1996.

[6] R. B. Dannenberg, W. P. Birmingham, G. P. Tzanetakis, C. P. Meek, N. P. Hu, and B. P. Pardo, "The musart testbed for query-by-humming evaluation," *Comput. Music J.*, vol. 28, pp. 34–48, June 2004.

[7] A. P. Klapuri, "Automatic music transcription as we know it today," *Journal of New Music Research*, vol. 33, no. 3, pp. 269–282, 2004.

[8] T. Tolonen and M. Karjalainen, "A computationally efficient multipitch analysis model," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 6, pp. 708 –716, nov 2000.

[9] A. Klapuri, "Multipitch analysis of polyphonic music and speech signals using an auditory model," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, pp. 255–266, 2008.

[10] H. Takeshima, Y. Suzuki, K. Ozawa, M. Kumagai, and T. Sone, "Comparison of loudness functions suitable for drawing equal-loudness level contours," *Acoust. Sci. Tech*, vol. 24, no. 2, pp. 61–68, 2003.

[11] C. Semal and L. Demany, "The upper limit of 'musical' pitch," *Music Perception: An Interdisciplinary Journal*, vol. 8, no. 2, pp. 165–176, 1990.

[12] K. Krumbholz, R. D. Patterson, and D. Pressnitzer, "The lower limit of pitch as determined by rate discrimination," *J. Acoust. Soc. Am.*, vol. 108, no. 3, pp. 1170–1180, 2000.

[13] D. Pressnitzer, R. D. Patterson, and K. Krumbholz, "The lower limit of melodic pitch," *J. Acoust. Soc. Am.*, vol. 109, no. 5, pp. 2074–2084, 2001.