Efficient Tempo and Beat Tracking in Audio Recordings*

JEAN LAROCHE

Creative Advanced Technology Center, Scotts Valley, CA 95066, USA

Automatic beat tracking consists of estimating the number of beats per minutes at which a music track is played and identifying exactly when these beats occur. Applications range from music analysis, sound-effect synchronization, and audio editing to automatic playlist generation and deejaying. An off-line beat-tracking technique for estimating a time-varying tempo in an audio track is presented. The algorithm uses an MMSE estimation of local tempo and beat location candidates, followed by a dynamic programming stage used to determine the optimum choice of candidate in each analysis frame. The algorithm is efficient in its use of computation resource, yet provides very good results on a wide range of audio tracks. The algorithm details are presented, followed by a discussion of the performance and suggestions for further improvements.

0 INTRODUCTION

Estimating the tempo of a musical piece is a complex problem, which has received an increasing amount of attention in the past few years. The problem consists of estimating the number of beats per minute (bpm) at which the music is played and identifying exactly when these beats occur. Commercial devices already exist that attempt to extract a musical instrument digital interface (MIDI) clock from an audio signal, indicating both the tempo and the actual location of the beat. Such MIDI clocks can then be used to synchronize other devices (such as drum machines and audio effects) to the audio source, enabling a new range of "beat-synchronized" audio processing. Beat detection can also simplify the usually tedious process of manipulating audio material in audio-editing software. Cut and paste operations are made considerably easier if markers are positioned at each beat or at bar boundaries. Looping a drum track over two bars becomes trivial once the location of the beats is known. A third range of applications is the fairly new area of automatic playlist generation, where a computer is given the task to choose a series of audio tracks from a track database in a way similar to what a human deejay would do. The track tempo is a very important selection criterion in this context, as deejays will tend to string tracks with similar tempi back to back. Furthermore, deejays also tend to perform beat-synchronous crossfading between successive tracks manually, slowing down or speeding up one of the tracks so that the beats in the two tracks line up exactly during the crossfade. This can easily be done automatically once the beats are located in the two tracks.

The tempo detection systems commercially available appear to be fairly unsophisticated, as they rely mostly on the presence of a strong and regular bass-drum kick at every beat, an assumption that holds mostly with modern musical genres such as techno or drums and bass. For music with a less pronounced tempo such techniques fail miserably and more sophisticated algorithms are needed.

This paper describes an off-line tempo detection algorithm, able to estimate a time-varying tempo from an audio track stored, for example, on an audio CD or on a computer hard disk. The technique works in three successive steps: 1) an "energy flux" signal is extracted from the track, 2) at each tempo-analysis time, several tempo and beat candidates are calculated, 3) a dynamic programming algorithm is used to determine the final tempo track and downbeat locations. These steps are described in the next section. We start with a brief overview of previous work.

An increasing number of contributions to the subject of tempo detection and beat tracking can be found in audio and music conferences and journals. Early work included that of Allen and Dannenberg [1], where the assumption was made that note onsets are readily available to the algorithm (for example, via MIDI) and which focuses on real-time performance. Scheirer [2] developed a technique for on-line tempo estimation (that is, with limited access to future samples), based on an array of low-order resonant

* Manuscript received 2002 October 21; revised 2003 January 28.

filters with resonant frequencies corresponding to various tempi, fed by a signal quite similar to the energy flux used in our algorithm. Brown's efficient autocorrelation-based technique [3] also assumes a MIDI input, which is unsuitable for our problem. Goto and Muraoka [4] describe fairly complex "multiagent" systems, where multiple tempo/beat hypotheses are explored in parallel. The systems make use of high-level musical concepts such as drum pattern matching or chord-change detection [5], but require a very powerful computer. By contrast with our algorithm, Goto and Muraoka's techniques are based on a note-onset extraction stage where discrete onset times are estimated from the signal, a potential weak point. Dixon's algorithms [6]–[8] are also based on the analysis of time differences between detected note onsets in the signal. An extensive bibliography of additional work on tempo detection and beat tracking can be found in [9].

1 DESCRIPTION OF THE ALGORITHM

1.1 Calculating the "Energy Flux" Signal

To estimate the tempo of the track, an algorithm should primarily use salient features from the audio, such as note onsets, note changes, and percussion hits, because beats tend to occur at these time instants. A simple way to do that consists of locating fast variations in the frequency-domain contents of the signal. This is usually better than using the energy of the time-domain signal because note onsets or percussion hits can be hidden in the overall signal energy by continuous tones of higher amplitude, such as bass notes. Frequency-domain processing allows detecting such events, even if they are of much lower energy than other continuous signals in the audio. The signal is analyzed using a short-time Fourier transform, that is, short segments of audio are extracted at regular instants, multiplied by an analysis window, and transformed into the frequency domain via a Fourier transform. Denoting by $x(n)$ the signal, t_i the frame time in seconds, F_s the sampling frequency, and N the size in samples of the analysis window $h(n)$, the short-time Fourier transform $X(f, t_i)$ at the normalized frequency f and frame i is

$$X\left(f, t_i\right) = \sum_{n=0}^{N-1} h\left(n\right) x\left(n + F_s t_i\right) \mathrm{e}^{-2\mathrm{j}\pi f n} . \qquad (1)$$

A suitable value for the window size is about 10 ms, and the hop size (the interval between two successive FFT analyses $t_{i+1} - t_i$) can be set to 10 ms (no overlap is needed). In the following steps only the magnitude of the FFT is used. For each FFT a nonlinear monotonic compression function $G(x)$ is applied to the bin magnitude, so high-frequency components (such as high hat hits) are not masked by higher amplitude low-frequency components (such as bass notes). A logarithmic function can be used, but such a function does not behave well around zero. A simple power function $G(x) = x^{1/2}$ can be used instead,[1] or an inverse hyperbolic sinus $G(x) = \text{arcsin}(x)$, which behaves like a logarithm for large values of x, and is well behaved around zero, but it has a slope of 1 at 0.[2] Since our goal is to locate areas in time and frequency where there is a sudden energy increase, a first-order difference from frame to frame is then calculated on the result. The results for all the bins are summed together, and the result is half-wave rectified to obtain a positive energy flux signal $E(i)$, which exhibits sharp maxima at transients and note onsets (Fig. 1),

$$\hat{E}\left(i\right) = \sum_{f=f_{\min}}^{f_{\max}} G\left(\left|X\left(f, t_i\right)\right|\right) - G\left(\left|X\left(f, t_{i-1}\right)\right|\right) \qquad (2)$$

$$E\left(i\right) = \begin{cases} \hat{E}\left(i\right) & \hat{E}\left(i\right) > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

where $f_{\min}$ and $f_{\max}$ control the range of frequencies over which the summation is carried out (typically from 100 Hz to 10 kHz). This signal is used in the subsequent stage to select tempo and downbeat location candidates.

1.2 Selecting Tempo and Downbeat Location Candidates

A least-squares approach is used to determine the best candidates at time t_a for the tempo and downbeat location. Under the assumption of a tempo R (in beats per seconds) and a downbeat location t, an expected energy flux signal $E_{R,t}(i)$ is defined, which represents our a priori knowledge of what $E(i)$ should look like. The Euclidian distance between the expected and the actual energy flux

[1] Thanks to Miller Puckette for this trick.
[2] Thanks to one of the reviewers for this suggestion.

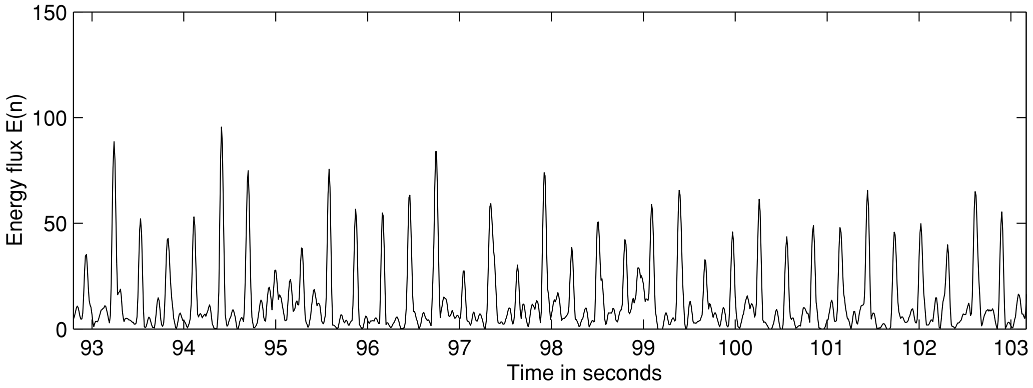

Fig. 1. Energy flux for a pop song with a strong beat.

signals $\Sigma_i[E(i) - KE_{R,t}(i)]^2$ can then be calculated, where K is an unknown scalar, which accounts for the fact that no assumption is made on the magnitude of $E(i)$. A small distance indicates a tempo and a downbeat location for which the expected and the observed energy flux signals match well, up to the gain factor K. The scalar K is obtained by minimizing the L2 norm and is easily found to be

$$K = \frac{\Sigma_{i=0}^{M-1} E(i) E_{R,t}(i)}{\Sigma_{i=0}^{M-1} E_{R,t}^2(i)} \tag{4}$$

where M is the horizon over which this distance is minimized. The squared L2 norm can then be expressed as

$$\sum_{i=0}^{M-1} \left[E(i) - KE_{R,t}(i)\right]^2$$
$$= \sum_{i=0}^{M-1} E^2(i) - \frac{\left[\Sigma \mathrm{E}(i) E_{R,t}(i)\right]^2}{\Sigma \mathrm{E}_{R,t}^2(i)}$$

and if $E_{R,t}(i)$ is properly normalized, so that $\Sigma_i E_{R,t}^2(i) = 1$ for all the values of R and t, we find the familiar result that the values of R and t that minimize the L2 norm are the ones for which the cross correlation $C(R, t)$

$$C(R,t) = \sum_{i=0}^{M-1} E(i) E_{R,t}(i) \tag{5}$$

is maximum. To calculate this cross correlation, the tempo is discretized into N_R values R_i (for example, from 60 to 150 bpm every 1 bpm), and for each tempo candidate, the downbeat location is also discretized into N_{D} equidistant times t_j^i. Given the analysis time t_{a} at which we need to estimate tempo candidates, and a candidate beat period $T_i = 1/R_i$ seconds, the candidate downbeat locations t_j^i are defined by

$$t_j^i = t_{\mathrm{a}} + T_i \frac{j}{N_{\mathrm{D}}}, \quad j = 0, \ldots, N_{\mathrm{D}} - 1 . \tag{6}$$

In other words, we consider downbeat locations every N_{D}th of the candidate beat period.

The expected energy flux signal E_{R_i,t_j^i} is simply chosen to be a series of discrete pulses, as depicted in Fig. 2. Choosing a discrete pulse signal makes the calculation of the cross correlation in Eq. (5) much less expensive, and does not compromise the accuracy of the results. Main pulses are located every T_i seconds, starting at the candidate downbeat location t_j^i. In addition, a series of secondary pulses is added for events occurring on half-beats. Finally a third series of pulses is added to account for events that occur on the first and third quarter-beats. The different pulse amplitudes reflect our expectation that the energy flux signal should be larger at certain beat subdivisions than at others. More will be said later on how these amplitudes can be chosen. Of course, the expected energy flux signals E_{R_i,t_j^i} can be computed and normalized in advance for each tempo and stored in memory.

Since we are interested in an estimate of the tempo around time t_{a}, the sum in Eq. (5) is limited to include the energy flux signal in the vicinity of t_{a}. The horizon can include 4 s or more. The longer the horizon, the clearer the estimate of the tempo and the downbeat location will be in the cross correlation, if the tempo is indeed constant over that duration. On the other hand, if the tempo varies over that horizon, the cross correlation might not give a good indication of the local tempo. Note that since the signal E_{R_i,t_j^i} in Eq. (5) is discrete, most of the products in the sum are null, except where $E_{R_i,t_j^i}(i)$ is nonzero. As a result it is much more efficient to implement Eq. (5) as a sum over the nonzero samples of $E_{R_i,t_j^i}(i)$.

At the end of this step we have a set of correlation values $C(R_i, t_j^i)$ for each tempo R_i and each corresponding downbeat candidate t_j^i. These correlation values can be normalized by their maximum value to give a local score $S(i, j)$ no larger than 1 for each candidate pair R_i, t_j^i,

$$S(i,j) = \frac{C\left(R_i, t_j^i\right)}{\max_{i,j} C\left(R_i, t_j^i\right)} . \tag{7}$$

It would be possible to use the complete set of pairs of candidates in the subsequent dynamic programming stage, but that choice would increase the computation cost unnecessarily. To avoid this, only the 10 to 15 best tempo candidates are kept, and for each of these candidates, only the best 10 to 15 downbeat locations are retained. Thus for each tempo candidate R_i, the best score $S_{\max}(i)$ over all the candidate downbeat locations is calculated,

$$S_{\max}(i) = \max_j S(i,j) . \tag{8}$$

$S_{\max}(i)$ is searched for local maxima, and the 10 to 15 largest local maxima define the 10 to 15 candidate tempi.

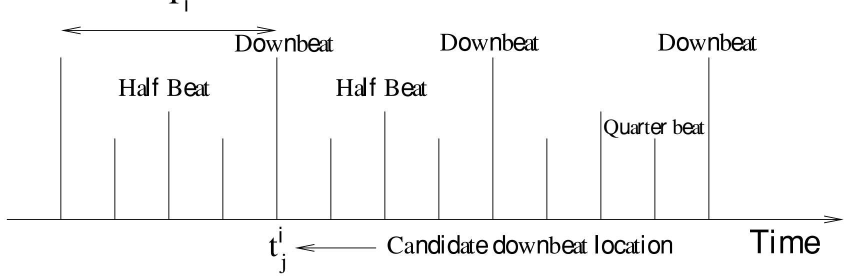

Fig. 2. Impulse signal to be correlated with energy flux.

Then for each of these pruned candidate tempi R_{i_k}, the 10 to 15 best candidate downbeat locations are selected according to the values of $S(i_k, j)$. This preselection yields 100 to 225 candidates for each analysis frame (instead of upward of 3200 if all the candidates were used). The local scores of these candidates are then used in the dynamic processing stage to be described. Fig. 3 gives an example of $S_{max}(i)$ for a rock song, analyzed over a 10-s horizon. Tempi were discretized every 0.2 bpm. The figure indicates a strong candidate at a tempo of 110.4 bpm. Other candidates would include 73.4 bpm, 82.6 bpm, and so on.

Fig. 4 shows the variations of $S_{max}(i)$ over time for 60 s of a Bach fugue (the last fugue in the Toccata in E Minor BWV 914 [10]). Dark areas indicate large values of $S_{max}(i)$. The moderately slow time evolution of the tempo around 120 bpm is quite visible.

1.3 Finding the Best Tempo Track and Downbeat Locations

At this point, for each analysis frame t_a (such as every second), we have a series of candidates for the tempo and the downbeat locations. The final step consists of going through the successive tempo analysis frames and finding in each frame the best candidates, according to a set of criteria to be defined. To that effect a dynamic programming technique is used [11]. To simplify the notations, the analysis frames will be indexed by a. At each frame, the set of N_a pairs of candidate tempi and downbeat locations will now be denoted by $\{R_i^a, t_i^a\}$, where R_i^a is the tempo value and t_i^a is the downbeat location for the ith candidate pair at frame a. A path is defined by a series i_a of selected candidates for each frame: $R_{i_a}^a$ is the selected tempo value

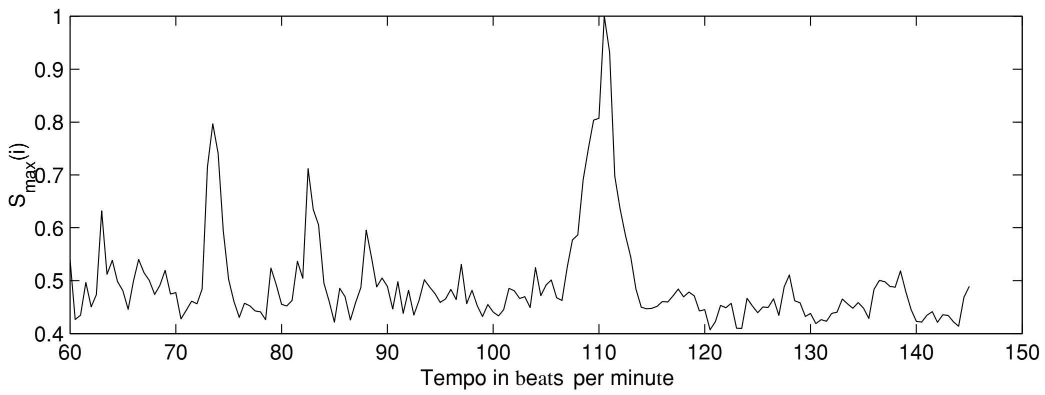

Fig. 3. $S_{max}(i)$ for a rock song.

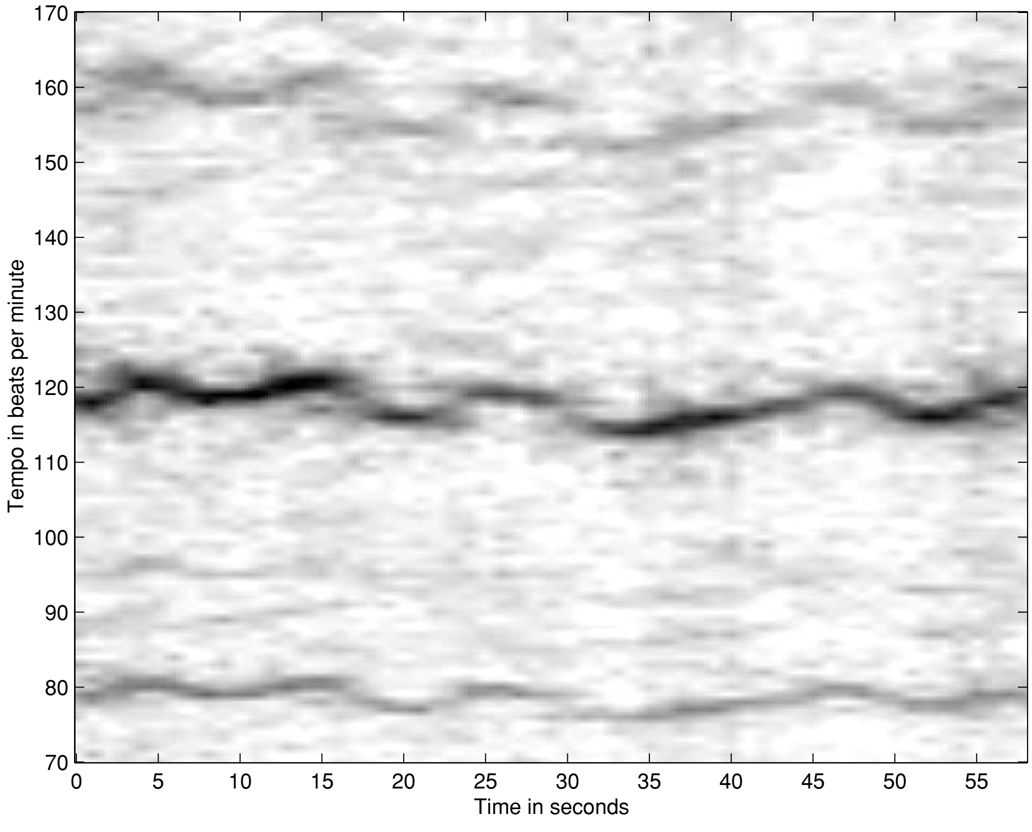

Fig. 4. Variations of $S_{max}(i)$ for a Bach fugue.

at frame a and $t^a_{i_a}$ is the selected downbeat location at frame a. The dynamic programming algorithm recursively defines a score $P(a, i_a)$ for a path arriving at candidate i_a at frame a, and requires that this score be a function of only three values:

- The score of the path at the previous frame $P(a - 1, i_{a-1})$, where i_{a-1} is the candidate through which the path goes at the previous frame
- The *local* score $S(i_a)$ of candidate i_a
- A *transition* score $J(i_{a-1}, i_a)$ measuring the cost of transitioning from candidate i_{a-1} at frame $a - 1$ to candidate i_a at frame a.

Mathematically we must have

$$P(a, i_a) = F\left(P(a-1, i_{a-1}), S(i_a), J(i_{a-1}, i_a)\right)$$

where $F(x, y, z)$ is any function used to combine the three scores. In the present case a simple sum is used, $F(x, y, z) = x + y + z$, but this is not a requirement.[3]

The dynamic programming algorithm allows us to find the path with the best score in an efficient manner, that is, without having to do an exhaustive test of all the possible paths (which would be prohibitively expensive). Only $N_{a-1}N_a$ evaluations of function F are required at each frame, which is quite reasonable in practice. An outline of the algorithm is given in the Appendix.

For our tempo estimation problem, Eq. (7) is used for the local score, and the transition score is defined to give good (large) scores to paths that have a smooth tempo and for which the downbeat locations are consistent with the tempo. The reasoning is as follows. In standard music the tempo varies slowly in time, with possible but rare sharp jumps, and this a priori knowledge should be reflected in our choice of the transition score. Furthermore, given a local tempo R in beats per second, consecutive downbeat locations should fall roughly every $1/R$ seconds. This should be reflected in the transition score as well. Given a candidate $\{R^i_{a-1}, t^i_{a-1}\}$ at frame $a - 1$ and a candidate $\{R^i_a, t^i_a\}$ at frame a, the transition score can therefore be defined as

$$J(i, j) = -\alpha M\left(R^j_a - R^i_{a-1}\right) - \beta \left|\operatorname{dist}\left(t^j_a - t^i_{a-1}, \frac{1}{R^j_a}\right)\right| \tag{9}$$

where dist(x, y) calculates the distance between x and the closest integer multiple of y,

$$\operatorname{dist}(x, y) \equiv x - y \cdot \operatorname{round}\left(\frac{x}{y}\right)$$

with round $(z) \equiv$ floor$(z + 1/2)$ being the integer nearest to z. $M(x)$ is a nonlinear positive function, and α and β are two positive constants that control the respective weights of each term. $M(x)$ can be chosen as

$$M(x) = \begin{cases} 0, & \text{for} |x| < \Delta R \\ 1, & \text{for} |x| \geq \Delta R \end{cases} \tag{10}$$

in which case the first term in Eq. (9) is 0 when the tempi at frames a and $a - 1$ are close enough, but becomes negative when they are significantly different. $G(x)$ allows the tempo to fluctuate within a small range $\pm\Delta R$ and imposes a fixed cost if the tempo jumps to a new value, because jumping to a very different tempo is no more unlikely than jumping to a closer tempo, if there is a tempo jump. The second term checks that the elapsed time between the two downbeat locations $t^j_a - t^i_{a-1}$ is close to a multiple of the beat period $T = 1/R^j_a$ and penalizes transitions for which this is not true. In practice it is better to give the downbeat location some flexibility (especially if downbeat locations are coarsely quantized). The second term in Eq. (9) can be modified accordingly. Fig. 5 shows the output of the dynamic programming algorithm for the Bach track used in Fig. 4.

The time-varying tempo values in Fig. 5 were calculated based on the time difference between successive beats in the path selected by the dynamic programming algorithm. The tempo track follows very well the gentle tempo variations of the performance, and is able to track a relatively sudden rallentando about 40 s into the piece.

[3] $F(x, y, z)$ can even be frame dependent; the same is true for the transition cost $J(\,)$.

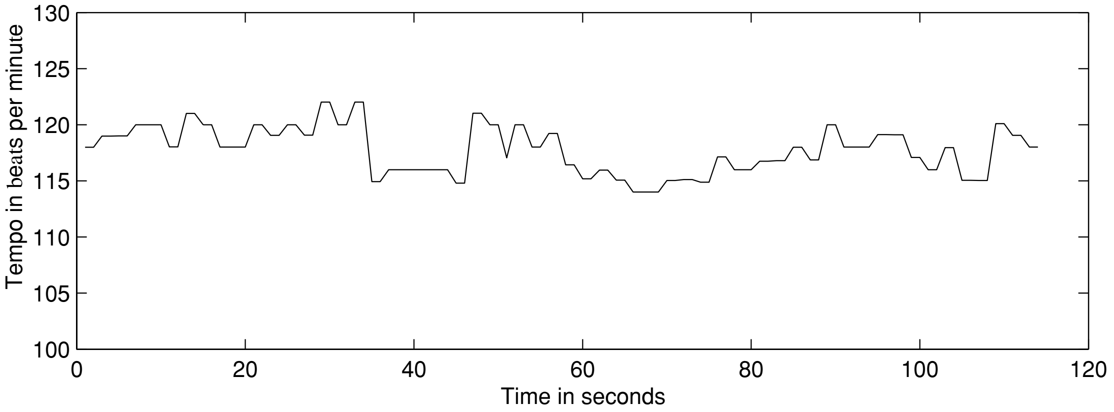

Fig. 5. Tempo track for Bach fugue used in Fig. 4.

2 ADJUSTING THE ALGORITHM PARAMETERS

2.1 Adjusting the Pulse Amplitudes

As mentioned before, the expected energy flux signal $E_{R,t}$ given the tempo R and the downbeat location t is a series of discrete pulses of unequal amplitudes. The pulse amplitudes can be determined by analyzing audio tracks with known tempo and downbeat locations, and calculating the average values of the energy flux at half-beat or quarter-beat times relative to the values at beat times. Table 1 presents the results of such an analysis on a series of seven tracks corresponding to seven different genres. The table was assembled using a limited number of tracks, and shows significant variability from genre to genre. (There are also significant variations from track to track within a genre.) The results should not be viewed as "hard" data, but nonetheless help reveal trends. For all the genres except latin and reggae, the downbeats collect the largest average values, with the half-beats receiving the second largest average values. This is consistent with the intuitive (but simplistic) idea that salient musical events fall primarily on the downbeat, then on the half-beat, and much less often on quarter-beats. By contrast, latin music consistently exhibits large values on the third quarter-beat, often larger than on the downbeat. Careful listening indeed reveals that many percussion accents tend to fall right before the downbeat (that is, on the third quarter-beat). Reggae also exhibits a very strong half-beat (often stronger than the downbeat), caused by the consistent snare-drum hits and guitar chords on the half-beat.

This primitive analysis helps us select reasonable values for $E_{R,t}$. A unity amplitude is selected for the downbeat, an amplitude of 0.65 for the half-beat, and of 0.2 for the quarter-beats. These settings are appropriate for various genres (rock, pop, techno, dance, and r&b among others) but are likely to yield wrong downbeat estimates for other genres such as latin or reggae.

2.2 Adjusting the Dynamic Programming Parameters

The parameters α and β in Eq. (9) can be adjusted by trial-and-error, using a large set of tracks, although this is a somewhat tedious task. A better solution consists of systematically testing a large number of parameter pairs for a large set of tracks of known tempo and downbeat locations. For each track we find the (hopefully not empty) region in (α, β) that gives the expected results. Any choice of parameters within the intersection of all such regions will yield accurate results for all the tracks tested. Of course, there is no guarantee that the intersection will not be empty, and that the chosen parameters will yield accurate estimates for other tracks not in the training set. Fortunately it was found in practice that the dynamic programming stage is quite robust in that regard. Fairly large relative parameter variations (such as multiplicative factors of 0.2 to 5) still yield accurate results.

Table 1. Average relative amplitudes of downbeat, first quarter-beat, half-beat, and third quarter-beat for various genres, in the energy flux signal $E(R, t)$.

Genre	0	1/4	1/2	3/4
Techno	1.0	0.1	0.6	0.4
Pop	1.0	0.1	0.8	0.2
R&b	1.0	0.2	0.5	0.2
Rock	1.0	0.2	0.7	0.2
Reggae	1.0	0.3	1.1	0.5
Latin	1.0	1.0	0.9	1.5

3 RESULTS AND COMMENTS

As mentioned in several papers on the subject [9], [6], a formal evaluation of a beat-tracking algorithm is not an easy task to complete, because of the lack of available "beat-labeled" audio tracks, the inherent ambiguity in the definition of musical beat, and the lack of a "standardized" set of test tracks. (It is always possible to test an algorithm on tracks for which it performs well.) Beat-tracking and tempo-detection algorithms can make roughly three kinds of errors: 1) a wrong tempo is estimated, 2) the downbeat is placed at the wrong beat subdivision, and 3) the downbeat is slightly off (by a few percent of the beat period). Informal tests show that the third type of error (slight misestimation of the downbeat locations) can be minimized to an arbitrary level if the energy flux signal is sampled fast enough and if enough downbeat candidates are selected prior to the dynamic programming stage. By contrast, the gross tempo detection and downbeat misplacement errors 1) and 2) are much more difficult to avoid. These are the kinds of errors we focus on in this paper, as they truly reflect the practical reliability and usability of a tracking algorithm.

3.1 Common Types of Errors

3.1.1 Tempo Octave Errors

Tempo octave errors (by analogy with pitch detection), that is, an estimated tempo twice on one-half the "true" tempo, do not necessarily matter, depending on the music meter. For example, it is correct to tap every quarter-note in a 4/4 meter, or every half-note (at half the previous tempo), as long as the "right" half-note is selected (see the following). In fact there may not be an absolute "true" tempo in that case. However, while tapping every quarter-note in a 3/4 meter is correct, tapping every half-note is not (because that would be tapping the first and third beats of measure 1, then the second beat of measure 2, and so on). The same is true for time signatures such as 6/8, 12/8, and so on. This, unfortunately, is a common problem with beat-tracking algorithms, and this one is no exception. This is probably why most beat-tracking algorithms are tested on 4/4 music [9] (although it is also true that most contemporary rock/pop/dance/techno music is in 4/4). An example of such a track is Billy Joel's "Piano Man" [12], a 3/4 time signature for which the quarter-note is at about 175 bpm.

3.1.2 Downbeat Errors

The algorithm presented in this paper does not attempt to locate measure boundaries (that is, estimate which downbeat is the first downbeat in the measure), as this would require estimating signal features more musically meaningful than our simple energy flux (chord changes would be good candidates [5]). Common downbeat errors include placing the downbeat one eighth-note or one quarter-note (if the detected tempo corresponds to half-notes) after or before the true downbeat. In our algorithm, the culprit for such mistakes is usually the assumption that is made on the shape of the expected energy flux, given a tempo and a downbeat. In rock music it is not rare for the half-note to dominate the downbeat, because of the ubiquitous snare-drum hits on 2, in which case our algorithm might select to tap beats 2 and 4, rather than 1 and 3.

3.2 Performance of the Algorithm

The algorithm presented in this paper works very well on a vast range of tracks, especially those that contain percussion instruments, or instruments with fairly sharp attacks (such as piano or string pizzicati), provided the tempo does not vary too rapidly. The dynamic programming stage is able to manage moderately slow or progressive tempo changes[4] (for example, a tempo that doubles in the span of 30 s) as well as abrupt tempo changes[5] (the tempo switches instantly from 150 to 110 bpm). The algorithm also handles tempo "gaps" very well (tracks where the music stops for as long as 10 s), thanks to the overall optimality of the dynamic programming stage. Broadly speaking, the algorithm yields inadequate results on two main types of tracks: tracks with a very flexible tempo (such as classical music played rubato) and tracks that lack clear transients or note onsets. Most expressive performances of pieces from the classical repertoire provide good examples of tracks that are difficult to beat-track. In particular, rubato playing proves quite difficult to track because beat durations can vary by very large amounts in the span of a few beats. For example, it is not uncommon at the end of a musical phrase to see the tempo drop by as much as 50% in the span of a few seconds. The current algorithm does not track such changes well for two main reasons. First the tempo is assumed to be constant over the "match horizon" in Eq. (5), an assumption that is violated in such cases. This will tend to smooth out $S_{\max}(i)$ and blur its local maxima, possibly causing erroneous tempo and downbeat candidates to be selected for the dynamic programming stage. Second the algorithm allows notes to fall on any subdivision of the beat (quarter, eighth, or sixteenth notes) while attempting to keep the tempo somewhat smooth. This means the dynamic programming stage is more likely to pick a fairly constant tempo with note onsets cycling through successive beat subdivisions than a fast varying tempo with onsets falling on downbeats. Following such rapid tempo variations is known to be a difficult task [1].

Signals that lack clear transients or note onsets cannot possibly be tracked accurately, because the energy flux signal does not exhibit maxima at beat times. Examples include pieces played legato and voice-only tracks.[6] In that case a better front end would be needed, able to extract more musically meaningful features such as pitch or chord changes (as suggested in [5]).

From a computation point of view, the algorithm is moderately costly, running about 50 times faster than real time (on a Pentium III at 850 MHz, for a 44.1-kHz wave file, computing the tempo every second, with an analysis window size of 8 ms, a hop size of 8 ms, a tempo search range between 70 and 160 bpm every bpm, 30 downbeat candidates per tempo candidate, and keeping 225 pairs of tempo and downbeat candidates in the dynamic programming stage).

4 FUTURE WORK AND CONCLUSION

The algorithm presented in this paper yields very good results on rhythmic contemporary music such as pop, rock, dance, and techno, but it is less successful when the rhythm is less pronounced or when the tempo is extremely variable. In both cases, results should improve with an improved front-end analysis able to better extract meaningful musical events. Note, however, that systems that have an ideal front end where all and only the relevant musical events are used as input data (such as systems that use MIDI input) still have difficulty tracking flexible tempi [1]. An issue not addressed in this paper is that of determining bar boundaries, which can be important in practice. One application is the automatic segmentation of songs into their parts (intro, chorus, versus) since such subdivisions tend to fall on bar boundaries. Also, in the context of beat mixing (transitioning from one track to the next by aligning the beats via time scaling and crossfading), it is important to align not just the beats but also the bar boundaries for the transition to sound good. The energy flux signal $E(i)$ used in this paper is probably too primitive a feature to enable such a task. In fact, it is very instructive to listen to the energy flux signal (for example, by amplitude modulating a constant-frequency sinusoid with $E(i)$). This experiment reveals how little information is contained in the energy flux signal (at least, that can be processed by a human brain)—enough in most cases to detect the beat, but not much more than that. Future work will attempt to identify and extract better signal features for the beat-tracking task, and address the problem of locating bar boundaries.

Finally, the algorithm can be modified for on-line analysis, where the tempo is estimated in real time with very limited access to the future of the signal (for example, less than 500 ms). This is the subject of ongoing work and will be reported in a future paper.

Sound examples can be found at www.atc.creative.com/users/jeanl.

5 REFERENCES

[1] P. Allen and R. Dannenberg, "Tracking Musical Beats in Real Time," in *Proc. Int. Computer Music Conf.* (San Francisco, CA, 1990), pp. 140–143.

[2] E. D. Scheirer, "Tempo and Beat Analysis of Acoustic Musical Signals," *J. Acoust. Soc. Am.*, vol. 104, pp. 588–601 (1998 Jan.).

[4] As in the track "You Want It Back" [13].
[5] As in the track "Tubthumping" by Chumbawanba [14].
[6] However, this algorithm can track rhythmic solo vocal tracks accurately, such as the ubiquitous "Tom's Diner" song by Suzanne Vega.

[3] J. C. Brown, "Determination of the Meter of Musical Scores by Autocorrelation," *J. Acoust. Soc. Am.*, vol. 94, pp. 1953–1957 (1993 Oct.).

[4] M. Goto and Y. Muraoka, "Music Understanding at the Beat Level — Real-Time Beat Tracking for Audio Signals," in *Proc. IJCAI-95 Workshop on Computational Auditory Scene Analysis* (1995), pp. 68–75.

[5] M. Goto and Y. Muraoka, "Real-Time Rhythm Tracking for Drumless Audio Signals — Chord Change Detection for Musical Decisions," in *Proc. IJCAI-97 Workshop on Computational Auditory Scene Analysis — Int. Joint Conf. on Artificial Intelligence* (1997), pp. 135–144.

[6] S. E. Dixon, "A Beat Tracking System for Audio Signals," in *Proc. Conf. on Mathematical and Computational Methods in Music* (Vienna, Austria, 1999), pp. 101–110.

[7] S. E. Dixon and E. Cambouropoulos, "Beat Tracking with Musical Knowledge," in *Proc. 14th Eur. Conf. on Artificial Intelligence (ECAI 2000)* (Amsterdam, The Netherlands, 2000), pp. 626–630.

[8] S. E. Dixon, "An Interactive Beat Tracking and Visualisation System," in *Proc. Int. Computer Music Conf.* (Havana, Cuba, 2001).

[9] M. Goto and Y. Muraoka, "Issues in Evaluating Beat Tracking Systems," in *Working Notes of the IJCAI-97 Workshop on Issues in AI and Music* (1997).

[10] J. S. Bach, "Glenn Gould" Bach, the Toccatas and Inventions," CBS Masterworks M2k42269 (1986).

[11] J. F. Silverman and D. P. Morgan, "The Application of Dynamic Programming to Connected Speech Recognition," *IEEE ASSP Mag.*, vol. 7, pp. 6–25 (1990 July).

[12] B. Joel, "Greatest Hits," vols. I and II, Sony (1998).

[13] Propellerheads, "Decksanddrumsandrockandroll," Dreamworks (1998).

[14] Chumbawanba, "Tubthumper," Universal (1997).

APPENDIX DYNAMIC PROGRAMMING ALGORITHM

This appendix gives an outline of the successive steps in a general dynamic programming algorithm. More details can be found, for example, in [11]. The algorithm goes as follows.

1) Initialize the best scores $\hat{P}(0, i) = p_i$ for each of the candidates i in the first frame.

2) At frame $a = 1$, calculate the scores of the best paths arriving at candidate 0 of frame a, coming from candidate k in the previous frame, using

$$R(k) = F\left(\hat{P}(a-1, k), S(0), J(k, 0)\right) \qquad (11)$$

where $\hat{P}(a-1, k)$ denotes the score of the best path arriving at candidate $(a-1, k)$. Do that for $k = 0, 1, \dots, N_{a-1} - 1$, where N_{a-1} is the number of candidates in frame $a - 1$.

3) Find the candidate $k_{\max}$ for which $R(k_{\max})$ is maximum. Set

$$\hat{P}(a, 0) = R(k_{\max})$$

and keep track of the candidate k this path originated from at frame $a - 1$. We now know the best path arriving at candidate 0 at frame a.

4) Repeat steps 2 and 3 for all the candidates 1, 2, … , $N_a - 1$ in frame a.

5) Go to the next frame and do steps 2, 3, and 4, and so on.

6) At the last frame I, pick the candidate with the best score $\hat{P}(I, i)$. This indicates the end of the best path (the path with the best score). Then backtrack to retrieve the candidates this path came from at frame $I - 1, I - 2, \dots$.

This algorithm only requires $N_{a-1}N_a$ evaluations of function F at each frame.

THE AUTHOR

Jean Laroche was born in Bordeaux, France, in 1963. He received an M.S. degree from the Ecole Polytechnique in 1986 and a Ph.D. degree in signal processing from the Ecole Nationale Supérieure des Télécommunications, Paris, in 1989.

In 1990 he was a fellow of the ITT international grant at the Center for Music Experiment, University of California at San Diego. In 1991 he became an assistant professor in the Signal Department at Telecom Paris, teaching audio and speech processing and acoustics. Since 1996 he has been a principal scientist at the Creative Advanced Technology Center in Scotts Valley, CA, helping to design techniques for advanced music and audio processing.