Real-time Phase Vocoder Manipulation by Runner's Pace

Jason A. Hockman[1] **Marcelo M. Wanderley**[2] **Ichiro Fujinaga**[1]
Centre for Interdisciplinary Research in Music Media and Technology (CIRMMT)
[1]Distributed Digital Music Archives and Libraries (DDMAL)
[2]Input Devices and Music Interaction Labratory (IDMIL)
McGill University, Montreal, QC, Canada
jason.hockman@mail.mcgill.ca, marcelo.wanderley@mcgill.ca, ich@music.mcgill.ca

Abstract

This paper presents a method for using a runner's pace for real-time control of the time-scaling facility of a phase vocoder, resulting in the automated synchronization of an audio track tempo to the generated control signal. The increase in usage of portable music players during exercise has given rise to the development of new personal exercise aids, most notably the Nike+iPod system, which relies on embedded sensor technologies to provide kinematic workout statistics. There are also systems that select songs based on the measured step frequency of a runner. The proposed system also uses the pace of a runner, but this information is used to change the tempo of the music.

Keywords: NIME, synchronization, exercise, time-scaling.

1. Introduction

During exercise, people listen to music for a wealth of reasons, e.g., as a distraction from discomfort, for reduction of tension, or to block repetitive noises made during exercise [1]. Music is also used for pacing information, as people often synchronize their motions with the period of an audio source. Tracks are selectively chosen for their tempo and perceived energy, and act as sources of motivation during exercise, thereby providing longer workout routines with less effort [2].

The popularization of portable music players for audio playback, as well as additional onboard functionalities, such as bluetooth, make these devices ideal for the development of exercise aids. Apple's Nike+iPod package [1] offers running shoes equipped with sensors that connect to an iPod via bluetooth. Although the transmitter sends data used for the determination of pace and exercise statistics, currently this information is not used to adaptively adjust the music playback tempo or track selection of the player.

In-shoe sensor systems have been well documented in the literature (for an overview, please refer to [3]). [4] incorporates force-resistant sensors (FSRs) to detect walking patterns, while [5] utilizes embedded FSRs, accelerometers, as well as pressure, bend and other sensors, towards the development of a musical controller. In the biomedical field, [6] has embedded both pressure/temperature sensors and a humidity sensor to analyze the effects of plantar pressures, temperature, and humidity on the breakdown of skin in diabetic patients, and [7] presents a low-cost system for gait monitoring and generation of kinematic data. More recently, [8] seeks to utilize a runner's changing step frequency information derived from in-shoe accelerometer data to control an automated track selection process during exercise.

We present a system for the real-time automated tempo synchronization of an audio track to a runner's step frequency using a method of control similar to those found in the parametric control of digital audio effects [9], and dynamic conducting strategies [10]. The result is an adaptive musical accompaniment for a runner, which closely follows step frequency fluctuations, leading towards a more cohesive relationship between a runner and music player.

The remainder of this paper is structured as follows. In §2, we outline the system hardware and design considerations. In §3, we present the sensor parsing and runner's pace estimation processes. An explanation of adaptive time-scaling with the phase-vocoder is given in §4. The method of communication between the pace estimation and time-scaling is described in §5, and discussion follows in §6.

2. System Design

An overview of the system is presented in Figure 1. An Analog Devices ADXL203EB two-axis accelerometer has been fitted on the rear-right position on a running shoe. This positioning was determined to provide the largest differential for two-axis acceleration data between extreme points of stride. In our prototype system (seen in Figure 2), X and Y axis information is sent by wires to analog inputs on an Arduino Diecimila USB input/output board, which has been fastened to a project box modified to fit on a belt.

The Diecimila sends messages to a computer via USB

[1] http://www.apple.com/ipod/nike

NIME09, June 3-6, 2009, Pittsburgh, PA

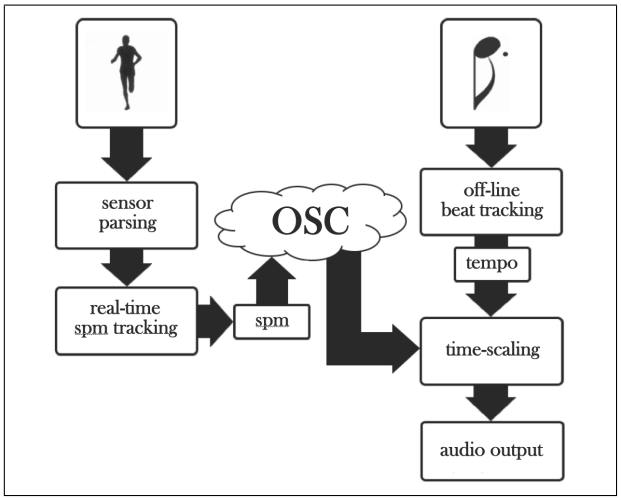

Figure 1. Overview of system design. Footsteps are detected by a sensor and steps per minute (SPM) are assessed. The SPM calculation is sent to a time-scaling program via Open Sound Control network. Audio tempo and SPM are input in the time-scaling stage for audio signal transformation.

connection. As they are received, messages are pre-processed. Next, we extract the approximate beat period from this data, and estimate step frequency. These values are then sent across an Open Sound Control network to a phase vocoder implementation that anticipates these messages. Here the step frequency message is combined with a pre-existing audio tempo message, for creation of a time-scaling rate which affects output audio tempo, without adjusting the pitch. The implementation is written in C and available for download [2] (the phase vocoder relies on the WaoN framework). [3]

Figure 2. Prototype shoe/sensor system. The accelerometer is attached to the right-rear position of the shoe. Data passes from the sensor to the I/O board mounted in a project box.

3. Sensor Data Parsing

To track the steps per minute (SPM) of a runner, the system must first locate the frequency of particular events understood as footsteps within a continuous signal.

[2] www.music.mcgill.ca/∼hockman/projects/beat-to-the-run
[3] www.kichiki.com/WAON/

An initial strategy that comes to mind might be rooted in the localization of discrete events, as utilized in several audio onset detection methods (see [12] for an overview). From such a representation, a calculation of inter-onset intervals (IOI) might then be achieved, followed by a histogramming or averaging method for the generation of a step period. While this is a feasible approach, undetected or spurious onsets could potentially skew the SPM calculation. The proposed method is to abandon the expectation of localized discrete events, and to generate SPM from the continuous sensor data. This has the added benefit of making the calculation resilient to spurious events, and thus robust against smaller changes in SPM. This technique is similar to existing beat tracking methods [13, 14, 15], but has been optimized for the input signal specific to the task. Below is an overview of the method (also presented in Figure 3).

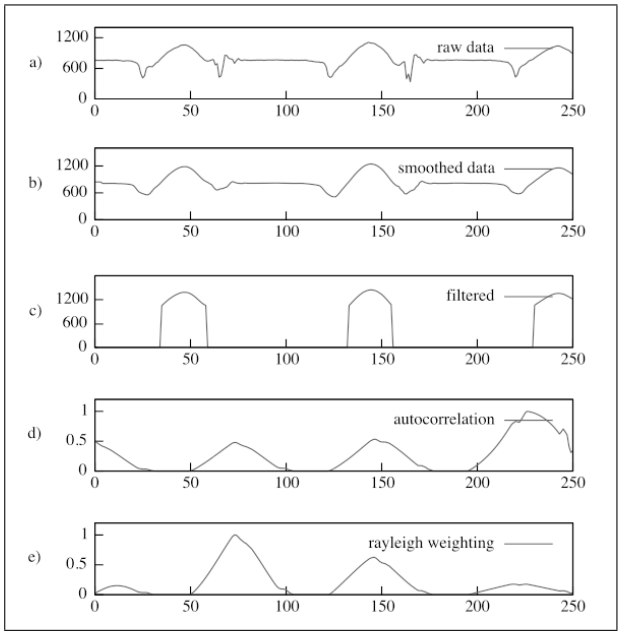

Figure 3. Sensor parsing and tempo estimation. (a) input signal across frame i (length $= 256$ samples); (b) adaptive smoothing of frame i; (c) sample-zeroing below adaptive average; (d) normalized autocorrelation; (e) autocorrelation with Rayleigh distribution weighting. Frame period δ_i is found by finding the maximum value of (e).

The two-axis accelerometer signal is returned to the Arduino board and is sent via USB (at 9600 baud) to the computer at a sampling period $s_{res} = 10$ ms, where it is summed and placed in a sliding-window buffer of size N. Several window and hop sizes were tested, and $N = 256$, and hop $= \frac{N}{16}$ were determined to offer optimal results for window and sliding buffer sizes capable of supplying an accurate representation of longer periods, while being short enough to track acceleration/deceleration, and provide timely SPM calculations (approximately every 160 ms).

Once a buffer is filled (Figure 3.a), the input is adaptively filtered using a sliding median window (Figure 2.b), which effectively acts as a lowpass filter, by recalculating each sample n as the mean of itself, $\frac{m}{2}$ prior, and $\frac{m}{2} - 1$ subsequent samples ($m = 8$). In subsequent stages (Figure 3.d and 3.e), it is necessary to further exploit the significant amplitude generated upon impact. We therefore set all samples below the filtered buffer mean to zero (Figure 3.c).

Autocorrelation is then performed on the resultant signal (Figure 3.d), which produces time-lag peaks representing possible beat periods. Both realistic usage of the end application, as well as physical constraints of the human motor system, suggest that we may limit our search for a maximum value to those peaks whose corresponding tempi result in no lower than 40 SPM (Γ_L), and no greater than 200 SPM (Γ_U) [11]. A possible approach could be to focus our decision of a maximum value within these lags; however it is necessary to choose a single maximum peak from the autocorrelation vector, and a spurious choice could potentially throw off the tempo calculation. To minimize the possibility of incorrect peak selection, the autocorrelation is weighted with a Rayleigh distribution (Figure 3.e), whose central lag η is chosen as the period demonstrated to minimize step frequency octave errors (e.g., halving or doubling) as the runner accelerates or decelerates towards the upper or lower bounds of the acceptable range. The Rayleigh distribution [16] provides greater emphasis to periodicities found within the range of physical possibility, and is calculated as

$$\Omega_{RW}(n) = \frac{\tau}{\eta^2} e^{\frac{-\tau^2}{2\eta^2}}, \qquad \tau = n + 1. \tag{1}$$

The current frame period δ_i then is considered the maximum peak within the weighted autocorrelation vector, from which step frequency Γ_i is then returned by

$$\Gamma_i = 2 \cdot \frac{60}{\delta_i \cdot s_{res}}, \tag{2}$$

where the multiplication by two is due to the singular sensor in the current implementation. New SPM values must comply with $\Gamma_L < \Gamma_i < \Gamma_U$, otherwise $\Gamma_i = \Gamma_{i-1}$.

As in beat tracking, the most common error is the halving or doubling of SPM values given by an adjacent peak of the autocorrelation. We incorporate the following simple conditional statements to prevent against such errors:

1) if $\Gamma_i - \Gamma_{i-1} > \frac{\Gamma_{i-1}}{4}$, then $\Gamma_i = \frac{\Gamma_i}{2}$,
2) if $\Gamma_{i-1} - \Gamma_i > \frac{\Gamma_{i-1}}{4}$, then $\Gamma_i = 2 \cdot \Gamma_i$.

Finally, to protect against spurious SPM values, which are differentiated from long-term acceleration or deceleration trends by their transitory nature, desired audio tempo ξ_i is given by

$$\xi_i = \Gamma_{i-1} + \rho \cdot (\Gamma_i - \Gamma_{i-1}), \tag{3}$$

in which $\rho = 0.4$, a value determined to minimize outlier effects while still capable of following trends closely in informal acceleration and deceleration tests.

4. Time-Scaling

We now turn our focus to the phase vocoder implementation [17], which has been adapted to iteratively receive step frequency messages, and adjust the output audio waveform accordingly. The phase vocoder is an optimal tool for time-scaling, as it is capable of individual manipulation of temporal and spectral information of a signal. Moreover, the STFT version has been proven capable of high-quality time and pitch modifications in a timely fashion [18].

The phase vocoder performs time-scale transformations through modification of the ratio between analysis and synthesis hop sizes. To reduce the length of an input signal, successive IFFTs are placed closer together than the default 1:1 correspondence; for a longer duration, they are placed further apart. For instantaneous control of audio playback rate by ξ_i, analysis hop size ζ_a is updated by

$$\zeta_a = \frac{\xi_i}{\beta_A} \zeta_s, \tag{4}$$

in which β_A is the predetermined tempo of the audio signal, and ζ_s is the synthesis hop size. For stability of frequency values, phase information is also modified by the updated ζ_a. Resynthesis phase difference is provided by

$$\Delta\psi(k) = \frac{\zeta_s}{\zeta_a} \Delta\phi(k), \tag{5}$$

where $\Delta\phi(k)$ is the phase difference determined in the analysis stage for bin k across analysis hop interval of size ζ_a.

5. Communication

Step frequency calculations exist on a different timescale than does the phase vocoder transformation. Although efficiency has been significantly increased by using a sliding window buffer with a $\frac{N}{16}$ hop size, new SPM values are not guaranteed upon every analysis iteration of the phase vocoder. The decision to use a non-blocking network solution became apparent after several attempts to integrate the SPM calculation into the phase vocoder architecture, without disrupting the timely audio processing essential for a continuous audio stream.

Communication between the two programs is handled by Liblo, [4] a lightweight implementation of Open Sound Control, which provides simple *lazy* communication between processes that potentially exist on different timescales (although on the same machine). In the presented program, SPM messages are made available by the server, while the client, the phase vocoder, may extract this information at its convenience. Once the phase vocoder has received the SPM message, it is reused until another is made available.

[4] http://liblo.sourceforge.net/

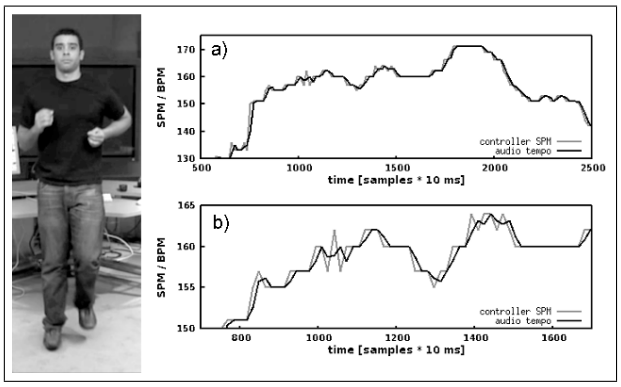

Figure 4. Example system performance. a) Audio track tempo (original tempo = 100 BPM) closely follows the control signal. New SPM values are determined approximately every 160 ms. b) A closer view demonstrates the resilience of the tempo following to sudden changes in SPM.

6. Discussion

While subjective evaluation has yet to be performed, preliminary results demonstrate that audio transformations sound as expected, with minimal disagreeable artifacts; a result due mainly to the efficiency of the phase vocoder implementation. For standard usage (i.e., walking, jogging, and running), the presented system functions proficiently. [5] Figure 4.a displays an example of recorded system performance. The audio tempo follows SPM values as they are produced; however, as is seen in Figure 4.b, sporadic changes in SPM are not translated directly to audio BPM, due to eq.3. As a result of the frequency of SPM calculations, audio tempo quickly adapts to actual acceleration (as in Figure 4.a between 700 and 800 ms), while capable of converging to a steady BPM (as in Figure 4.b between 1500 and 1700 ms).

To reduce the effect of spurious SPM calculations, the simple addition of a second sensor in the opposite shoe would increase the accuracy of SPM values by effectively doubling the number of footsteps within each analysis window, without increasing window length–a modification that would lead to both greater latency between SPM values and the reduction of estimation accuracy.

Currently, time-scaling relies on a single tempo for an entire track. A dedicated beat tracking technique would provide a means to affect accurate change not only as running pace is established, but as audio beat period is modified. This would also allow for synchronization of upcoming audio beat events with estimates of future steps. Parallel processes available through OSC's client-server relations offer encouraging prospects in this regard. In addition, we are currently investigating the inclusion of playlist selection and automated mixing functionalities to the presented method.

[5] http://music.mcgill.ca/~hockman/projects/beat-to-the-run/video.html

References

[1] M. Stevens and A. Lane,"Mood-regulating strategies used by athletes," *Athletic Insight: Journal of Sport Psychology*, vol. 3, no. 1, pp. 1–12, 2001.

[2] A. Beckett, "The effects of music on exercise as determined by physiological recovery heart rates and distance," *Journal of Music Therapy*, vol. 27, pp. 126–36, 1990.

[3] E. Miranda and M. Wanderley, *New Digital Music Instruments: Control and Interaction Beyond the Keyboard*, AR Editions, Middleton, Wisconsin, 2006.

[4] I. Choi and C. Ricci, "Foot-mounted gesture detection and its application in a virtual environment," *Proc. of 1997 IEEE International Conf. on Systems, Man, and Cybernetics*, vol. 5, pp. 4248–53, 1997.

[5] J. Paradiso, E. Hu, and K. Hsiao, "The Cybershoe: A wireless multisensor interface for a dancer's feet," *Proc. of International Dance and Technology '99*, pp. 57–60, 1999.

[6] R. Morley, E. Richter, J. Klaesner, K. Maluf, and M. Mueller, "In-shoe multisensory data acquisition system," *IEEE Transactions on Biomedical Engineering*, vol. 48, no. 7, pp. 815–20, 2001.

[7] S. Morris and J. Paradiso, "Shoe-integrated sensor system for wireless gait analysis and real-time feedback," *Proc. of the 2nd Joint EMBS/BMES Conf.*, pp. 2468–9, 2002.

[8] N. Masahiro, H. Takaesu, H. Demachi, M. Oono, and H. Saito, "Development of an automatic music selection system based on runner's step frequency," *Proc. of 2008 International Conf. on Music Information Retrieval*, pp. 193–8, 2008.

[9] A.M. Stark, M.D. Plumbley, and M.E.P. Davies, "Real-time beat-synchronous audio effects," *Proc. of the 2007 Conf. on New Instruments for Musical Expression*, pp. 344–5, 2007.

[10] J. Borchers, E. Lee, W. Samminger, and M. Muhlhauser, "A real-time audio/video system for interactive conducting," *ACM Multimedia Journal Special Issue on Multimedia*, vol. 9, no. 5, pp. 458–65, 2004.

[11] K. Hoffman, "Stature, leg length and stride frequency," *Track Technique*, vol. 46, pp. 1463–9, 1971.

[12] J.P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. Sandler, "A tutorial on onset detection in music signals," *IEEE Signal Processing Letters*, vol. 13, no. 5, pp. 1035–47, 2004.

[13] M.E.P. Davies, *Towards automatic rhythmic accompaniment*, PhD Thesis. Department of Electronic Engineering, Queen Mary, University of London, 2007.

[14] M. Alonso, B. David, and G. Richard, "Tempo and beat estimation of musical signals," *Proc. of 2004 International Conf. on Music Information Retrieval*, pp. 158–63, 2004.

[15] D. Ellis and G. Poliner, "Identifying cover songs with chroma features and dynamic programming beat tracking," *Proc. of the 2007 International Conf. on Acoustics, Speech and Signal Processing*, pp. 1429–32, 2007.

[16] A. Papoulis, *Probability, Random Variables, and Stochastic Processes, 2nd ed.*, New York: McGraw-Hill, 1984.

[17] "WaoN: A wave-to-notes transcriber," [Web site] 2006, [2008 Nov 20], Available: www.kichiki.com/WAON

[18] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Trans. on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–32, 1999.