基于自适应白化的音乐节拍实时跟踪算法 *

王 跃, 谢 磊, 杨玉莲
(西北工业大学 计算机学院 音频、语音与语言处理研究组, 西安 710072)

摘 要: 提出一种基于自适应白化的音乐节拍实时跟踪算法。在对音乐信号进行触发点检测之前,构建音乐信号短时傅里叶变换的当前频域峰值表,对各频段的频谱幅值进行自适应加权,使各个频段保持相似的动态范围,改善音乐信号起伏变化较大时触发点检测准确性低的问题,进而提高节拍实时跟踪效果。在 MIREX2006 标准测试库上的节拍跟踪实验表明,自适应白化的引入可整体提高 P-score,特别是对于起伏变化很大的音乐信号,节拍跟踪效果的提高非常明显。

关键词: 音乐节拍跟踪; 自适应白化; 触发点检测; 实时
中图分类号: TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2009)05-1676-03

Adaptive whitening for real-time music beat tracking

WANG Yue, XIE Lei, YANG Yu-lian
(*Audio, Speech & Language Processing Group, School of Computer Science, Northwestern Polytechnical University, Xi' an* 710072, *China*)

Abstract: This paper proposed an adaptive-whitening-based real-time algorithm for music beat tracking, introduced adaptive whitening into onset detection phase of a music beat tracker. It preprocessed the STFT frames by adaptively normalizing the magnitude of each bin according to a present spectral peak table. This allowed each magnitude bin to achieve a similar dynamic range over time and improved onset detection performance for music with spectral roll-off effects and strongly-varying dynamics. Experimental results on the MIREX 2006 dataset shows that the proposed adaptive-whitening-based beat tracker achieves superior performance and it can significantly improve the P-scores for music samples with strongly-varying dynamics.

Key words: music beat tracking; adaptive whitening; onset detection; real-time

0 引言

实时跟踪音乐节拍是人类的一种自然能力。人们在听音乐时会本能地识别音乐的节拍,经常边听音乐边踏脚、拍手或点头,跟着节拍舞动。利用计算机实现音乐节拍的实时自动跟踪具有广泛的应用。例如,在大型的演唱会中,舞台灯光效果往往能给演出锦上添花,歌手在投入演唱时,五光十色的灯光如果能随着音乐节拍自动打开和关闭,可以给观众带来视听的双重享受。机器人舞蹈是另外一个有趣的应用,机器人随着音乐的播放,利用接收到的音乐信号分析出音乐节拍,根据节拍强弱做出相应的舞蹈动作,与音乐同舞,其乐无穷。这些应用往往要求节拍提取算法具有实时性或因果性,即能根据已播放的音乐在短时间内预测下一节拍的时刻。

利用计算机进行节拍提取分为两个阶段。第一阶段是从输入信号中构建中层表示(middle-representation)或驱动函数(driving function)[1]。通常不能直接对输入的音乐信号进行节拍跟踪,一方面是因为原始信号中包含了大量的冗余信息;另一方面,音乐在主观听感上是由一系列的事件组成,节拍就是一种明显的事件,人们在听歌时,会跟踪间断的节拍事件而非连续的音乐旋律,因此需要对信号进行简化,突出事件出现的可能位置。第二阶段是从中层表示中提取节拍周期和进行节拍开始点检测,在音乐开始阶段假定节拍稳定出现,随着音乐的播放,一旦出现节拍周期变化很大的状况,算法应该能及时捕捉,反映到节拍的周期输出中。

现行的节拍提取算法大致可以分为:a)是对 MIDI 这种符号化记谱音乐进行节拍分析,应用面相对较窄[2];b)是直接处理波形音频(waveform)的非实时算法,如 Dixon[3] 的 BeatRoot 算法,它通过多代理机系统从产生的候选节拍中提取最有效的节拍;c)是处理波形音频的实时算法,如 Goto[4] 的方法能够实时跟踪多个层次的节拍,但它对输入有严格限制,要求音乐是稳定的 4/4 拍的节奏,且算法代价大、效率不高。国际音乐信息检索评测 MIREX[5],特别针对节拍提取开展年度竞赛,吸引了众多大学和机构的参与。

近来,Davies[6] 提出了一种基于上下文的节拍实时跟踪算法,该算法用最近 6 s 的音乐信号,预测将来 1.5 s 内音乐的节拍,具有较好的实时性。其中节拍周期和节拍开始点的提取过程相互独立,计算复杂度低。Davies 的实时算法对输入音乐的节拍没有特殊要求,但对起伏变化很大的音乐,节拍提取的准确性较差,因为其基于触发点检测的中层表示部分容易丢失大量高频信息,造成触发点检测错误,影响节拍周期提取的正确性。本文针对这一问题,在触发点检测之前引入自适应白化算法,通过对频谱能量进行自适应加权,改善起伏较大音乐触发点检测不准的问题,提高实时节拍跟踪效果。在 MIREX2006[5] 标准数据集上进行的实验证明了改进方法的有

收稿日期: 2008-07-13; **修回日期**: 2008-09-21 **基金项目**: 国家教育部高等学校博士点学科专项基金资助项目(20070699015)
作者简介: 王跃,男,硕士研究生,主要研究方向为音乐信号处理;谢磊,男,副教授,博士后,主要研究方向为多媒体信息检索、语音与语言处理(xielei21st@gmail.com);杨玉莲,女,硕士研究生,主要研究方向为多媒体信息检索、语音与语言处理.

效性。

1　触发点检测存在的问题

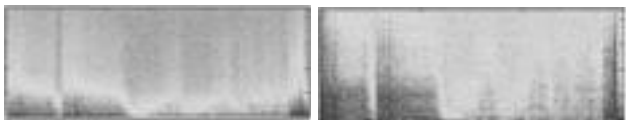
节拍自动跟踪的中层表示有多种方法,如基于触发点检测的方法[5]和多子带分解方法[7]等。触发点检测通常应用在音乐的节奏分析中,用来表示音乐事件和可能的音乐节拍点。触发点是指音乐信号的开始点,通常是信号幅值从零到达峰值的点。Scheirer 指出[7],节拍点一定是触发点,触发点不一定是节拍点,然而触发点为节拍提取提供了重要线索。触发点检测主要有基于频域能量、基于高频内容、基于相位和基于复频域的检测等方法[6]。Davies[6]的实验表明,不同类型的触发点检测方法对于节拍跟踪的效果影响较大。其中基于复频域差分的触发点检测方法效果最好,但该方法在音乐信号起伏变化很大,或信号低频段集中大量频谱能量导致 roll off 现象时,跟踪效果并不理想。

图 1(a)示意了《贝多芬第五交响曲》前 15 s 的波形,这段音乐信号开始的能量很高,之后急剧降低,又突然增强,后又急剧减小,这给触发点检测造成了很大的困难。其中对基于频域能量或者频谱能量的检测方法造成的影响较大。起伏变化很大的音乐信号,在进行短时傅里叶变换(short-time Fourier transform, STFT)之后会出现大量的范围变化很大的峰值。对 STFT 幅值进行限定阈值的归一化,可以在一定程度上提高触发点检测效果,但是对基于能量的触发点检测方法提高效果不明显[8]。从图 2(a)所示的同一段乐曲的频谱图中可以看出,低频占据了绝大部分频谱能量,低频段信息会“淹没”高频段信息(即 roll off 现象),从而导致基于复频域和高频内容的触发点检测方法失效。

另一方面,实时节拍分析存在一定的惯性,如果音乐开始的节拍跟踪不准,会引起连锁反应。对于开始能量很强的音乐,在节拍预测时,会默认当前信号的能量很强、节拍很规律,但是当能量随时间急剧衰减时,这样的变化通常不能被有效地检测,最终影响后续节拍检测的准确性。

(a)白化前　　(b)白化前

图1　《贝多芬第五交响曲》前15s的波形图



(a)白化前　　(b)白化后

图2　《贝多芬第五交响曲》前15s的频谱图

2　基于自适应白化的实时音乐节拍跟踪

2.1　自适应白化算法

为解决音乐信号起伏变化较大引起的触发点检测准确性低的问题,在触发点检测之前,引入 Stowell 等人[8]的自适应白化算法。每次从音频流读取 512 个采样点后,进行 512 点短时傅里叶变换。由于每个点都代表了相应的频段(bin),进行短时傅里叶变换后,保存目前可见的相应频段的最大值,称为当前频域峰值表(present spectral peak table, PSPT)。当进行下一帧信号的短时傅里叶变换之后,将各个频段与当前频域峰值表中的相应频段进行比对,若小于当前的频谱峰值轮廓相应频段,则进行增量相加并进行更新;若大于等于当前的频谱峰值轮廓,则更新为当前的频谱峰值轮廓。该方法可使各个频段保持相似的动态范围。

Stowell 的原始白化算法只采用前一帧的频谱值来表示当前频谱值(cur_mag = a × old_mag)。本文同时考虑了当前帧和前一帧的影响,将自适应加权修改为

$$\text{cur_mag} = (1 - a) \times \text{cur_mag} + a \times \text{old_mag} \tag{1}$$

其中:cur_mag 为当前帧频谱值;old_mag 为前一帧频谱值;a 为加权系数。实验表明,同时考虑前一帧和当前帧的效果较好。

《贝多芬第五交响曲》前 15 s 的波形经过自适应白化的效果如图 1(b)和图 2(b)所示。白化后的波形图与原波形图相比,在信号强和弱的部分均呈现出相似的动态范围。对于白化后的频谱图,在同一段时间内,高频和低频部分的能量差别显著减小,整个频谱能量分布变得较为平坦和均匀,之前提到的 roll off 现象已基本消除。

自适应白化算法伪代码如下:

```
Adaptive whitening(fft_magnitude)
输入:当前 STFT 变换后的 512 点幅值
输出:经过自适应白化后的 512 点幅值
  if frames = =1
  for each 1 < =i < =512
    if fft_magnitude(i) > threshold //经验值
      pspt(i) = fft_magnitude(i);
      else
      pspt(i) = threshold;
  end for
end if
for each 1 < =i < =512
    old_mag = pspt(i);
    cur_mag = fft_magnitude(i);
    if cur_mag < old_mag
      cur_mag = (1 - a) * cur_mag + a * old_mag;//a 是加权系数
    end if
    pspt(i) = cur_mag;
end for
for each 1 < =i < =512
fft_magnitude(i) / = max (pspt(i),threshold);
end for
```

2.2　算法特点

自适应白化算法具有以下特点:

a)自适应性。自适应白化是针对中层表示的问题提出的,它不同于对每一帧都进行线性加权的高频内容方法,又不同于信号的预加重处理,而是针对当前待处理帧之前的信号进行相应的动态调整和加权。

b)实时性。实时节拍跟踪必然要求在进行中层表示时也能实时进行加权,该算法只利用当前信号之前帧的信息进行加权,效率高,能够满足实时性要求。

c)低耗性。整个加权过程只需要一个保存当前的频域峰值表的数组。针对当前帧各个频率段的幅值进行频域峰值的更新,即可进行自适应加权。这样的时空代价符合节拍提取的低耗性要求。

3　基于自适应白化的实时节拍跟踪算法

Davies 等人[6]的节拍跟踪算法是基于普通模态和上下文

模态的双模态机制方法,有效地解决了节拍跟踪的惯性问题。算法首先对输入的音乐信号进行加窗和短时傅里叶变换,然后提取触发点。当音乐信号达到 1.5 s(128 帧)后,将其更新输入到缓冲分析窗(6 s),进行节拍周期提取和开始点检测。节拍周期提取和开始点检测由普通模态和上下文模态完成。上下文模态负责维持节拍的连续性;普通模态则负责检测节拍的变化。算法首先进入普通模态,对自相关进行瑞利先验加权和梳状滤波加权,提取周期和节拍开始点,之后检测所提取节拍周期的变化情况。如果连续三个节拍周期起伏较小,就认为音乐的节拍比较稳定,应该保持节拍的稳定性和连贯性,进入上下文模态,以上下文模态提取的节拍周期和节拍开始点,计算并输出节拍位置;否则仍然维持在普通模态,根据普通模态提取的节拍周期和节拍开始点计算并输出节拍位置。普通模态和上下文模态是通过不同的概率分布对自相关序列进行加权的,普通模态采用瑞利分布,上下文模态采用高斯分布。

本文将前述自适应白化算法引入到节拍跟踪中。自适应白化算法主要关注节拍跟踪算法的中层表示部分,在提取触发点之前,将强弱信号调整到相似的动态范围。在保持原算法剩余部分不变的情况下,首先对窗口化后的短时傅里叶变换进行自适应白化,再进行触发点检测,最后从中层表示中提取节拍周期和进行节拍开始点检测。这样便构成了基于自适应白化的实时节拍跟踪算法,整个算法流程如图 3 所示。

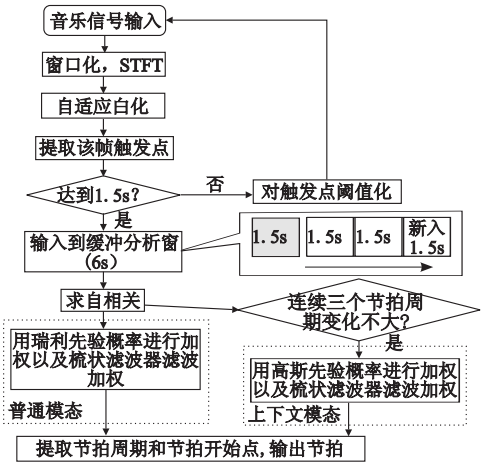


图3　基于自适应白化的节拍实时跟踪算法

4　实验

4.1　数据集

实验数据由 MIREX2006[5] 提供的公开测试样本构成,涵盖了不同风格、不同语言、不同节奏的音乐。每个样本时长 30 s,单声道,采样率为 44.1 kHz,量化精度 16 bit。每个样本由 40 位专业人士进行人工节拍标注,40 组标注好的节拍被当做正确的节拍,作为评判参照。

4.2　评测标准

评测标准采用 MIREX2006 中定义的 P-score。对每个音乐样本,忽略其前 5 s 的节拍,对其后 25 s 的节拍构建一个频率为 100 Hz 的脉冲,对每一组人工标注的节拍位置构建脉冲矢量 $a_{s[n]}$,S 表示第 1 ~ 40 组节拍,n 表示该组节拍的第 n 个拍子。然后用它与实际算法得出的节拍 $y[n]$ 进行互相关检验,称为 P-score,定义如下:

$$P = 1/s \sum_{s=1}^{S} 1/NP \sum_{m=-W}^{+W} \sum_{n=1}^{N} y[n] \cdot a_s[n-m] \tag{2}$$

其中:N 表示样本长度;NP 取样本和实际输出节拍更多的节拍数;窗口 W 用来描述一定的误差,定义为

$$W = \mathrm{round}(0.2 \times \mathrm{median}(\mathrm{diff}(\mathrm{find}(a_s)))) \tag{3}$$

P-score 越高,表示算法提取的节拍与人工标注越接近。

4.3　实验结果及探讨

考虑到实时性和低耗性,选取基于复频域差分、频谱能量和高频内容三种触发点检测方法对 20 个样本进行实验,加权系数和阈值设定为经验值 0.96 和 0.5。实验结果如表 1 所示。

对基于复频域的触发点检测表明,自适应白化后 P-score 比白化前有略微提高;对基于能量的触发点检测和基于高频内容的出发点检测,自适应白化后的 P-score 比白化前分别提高 2.8% 和 5.4% 左右。

对单个样本计算的 P-score 统计显示,自适应白化使 train8、train13 和 train15 三个样本的 P-score 提高较大,如表 2 所示。对波形和频谱图的观察表明,这三个样本也正是起伏很大的音乐样本。这验证了自适应白化对起伏较大音乐节拍跟踪的有效性。

表1　实时节拍跟踪实验结果(P-score)

触发点检测方法	非自适应白化	自适应白化
基于复频域	0.447 7	0.449 8
基于能量	0.432 8	0.444 8
基于高频内容	0.425 8	0.448 8

表2　起伏较大音乐的实时节拍跟踪实验结果(P-score)

音乐样本	非自适应白化	自适应白化
train8	0.598 2	0.761 4
train13	0.192 4	0.390 5
train15	0.278 6	0.391 6

另外,train1、train8、train9 和 train11 的 P-score 远远高于其他样本,这些样本音乐节奏性强且节奏适中,为 120 BPM 左右,是人类听觉比较倾向的节奏[6]。节奏偏快(如 train19 和 train20)或偏慢(如 train12)的音乐,其 P-score 普遍偏低,这是因为 Davies 算法普通模态中的瑞利先验分布是有偏的,即对于节奏为 120 BPM 左右的音频帧在加权时赋予较大值,而对其他节奏的音频帧赋值较小,尤其是对于偏快和偏慢的节奏,权值接近于零。这导致节奏较快和较慢的样本出现时,节拍跟不上或者节拍太快。

需要指出的是,本文实时节拍跟踪算法的 P-score 与 MIREX2006 评测中提供的算法有一定差距,排在其评测结果的第五名。然而 MIREX2006 评测中的所有算法均为非实时算法,本文算法具有较好的实时性,能够利用过去 6 s 的音乐信号,预测将来 1.5 s 内音乐的节拍。另外 Dixon[3] 的实验表明,MIREX2006 为每个样本提供了 40 个标注者提供的节拍标注,然而从这 40 个标注者提供的正确节拍中抽取节拍所计算的 P-score,取值在 0.34 ~ 0.73。也就是说,不同的人对于同一首音乐的节拍感觉也相差很大。

5　结束语

本文针对 Davies 实时节拍提取算法中中层表示存在的问题引入自适应白化方法,改善音乐信号起伏变化较大时触发点的检测效果,从而提高了节拍跟踪的准确性。在 MIREX2006 标准测试数据上的实验表明了改进算法的有效性。根据本文的算法,设计了一个音乐节拍实时跟踪演示系统,如图 4 所示。今后的工作包括:完善自适应白化算法,减少算法对加权系数和阈值设定这两个参数选择的依赖;针对原有的有偏先验分布估计,引入新的先验估计模型,进一步提高实时节拍跟踪算法的效果。

(下转第 1684 页)

器作为提供服务的 server。所有设备连接在 Cisco3750 交换机上,并分配内网 IP。设备的硬件配置如表 1 所示。

表 1　测试环境

设备	CPU	内存/MB	网络带宽/Mbps
SSG	Pentium IV 3.2 GHz	512	100
server 1	Pentium IV 2.4 GHz	512	100
server 2	Pentium IV 2.0 GHz	512	100
server 3	Celeron 2.4D	256	100
server 4	Celeron 2.4D	256	100

五台设备采用一样的操作系统 RedHat Linux 9,内核版本为 Linux Kernel 2.4.20,其他软件配置包括 gcc-3.2.2、glibc-2.3.2、GNU-ld-2.14 等。客户端也要安装 Tomcat5.0 和 MySQL 数据库,用于发布测试的服务。除了这些必需的配置外,系统进行了最小化安装,从而保证系统运行时的效率。充当 SSG 设备的服务器上安装实现的负载查询器 NECP server,并在内核中实现均衡调度模块,包含经典的 WRR 算法和本文中采用的 DWPS 算法,数据包转发采用 NAT 的方式。每个服务器节点上安装负载反馈器 NECP agent。

3.2　结果分析

测试包括三项内容:a)测试传统的静态 WRR 算法性能;b)测试不考虑服务类型和服务器负载能力的 DWPS 算法性能,即将负载能力向量和权重向量设成一样的值;c)测试完整的 DWPS 算法的性能。测试过程中,后台服务器提供两种服务(分别放到不同的目录下),即发布 10 KB 大小的静态页面,以及发布能够触发复杂运算的动态页面(JSP 文件)。结果如图 4 所示。

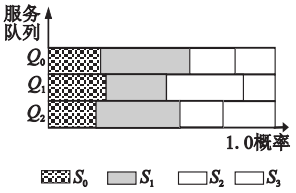


图3　转发概率空间

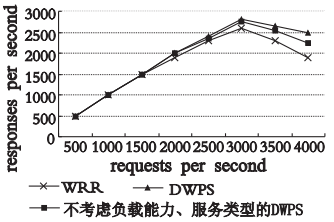


图4　测试结果

从测试结果中可以看出,在访问量不大的情况下,算法之间的区别并不明显。随着请求数量的增加,DWPS 算法的优势便逐渐体现出来了,并可得出以下结论:

a)带有动态反馈功能的 DWPS 算法比 WRR 更能够提高服务器系统的整体性能,使其具有更高的负载能力。

b)在考虑服务器的负载能力和服务类型的情况下,算法的负载均衡性能得到了一定的改善。

4　结束语

本文中实现的是一种基于服务类型动态反馈负载均衡算法,反馈机制采用 NECP 完成。该算法在仿真测试中体现出了一定的优越性,说明具有实际应用的价值,但是还存在需要进一步改进的地方。如何更加准确合理地设定权重向量,需要进一步的实验进行验证;在 SSG 设备上实现负载均衡模块的效率问题也将是进一步研究的内容。

参考文献:

[1] 洪英杰, 邵洪钢. 业务选择网关在增值业务系统中的应用[J]. 现代电信科技, 2005(4):32-34.

[2] CHEN Li-chuan, CHOI H A. Approximation algorithms for data distribution with load balancing of Web servers[C]//Proc of the 3rd IEEE International Conference on Cluster Computing. Washington DC:IEEE Computer Society, 2001:274-281.

[3] 章文嵩. Linux 服务器集群系统[EB/OL]. (2003-03-24)[2008-06-20]. http://www.linuxvirtualserver.org.

[4] CASSLICCHIO E, TUCCI S. Static and dynamic scheduling algorithm for scalable Web server farm[C]//Proc of the 9th IEEE Euromicro Workshop on Parallel and Distributed Processing. Mantova:[s.n.], 2001:369-376.

[5] IYENGAR A, MACNAIR E, NGUYEN T. An analysis of Web server performance[C]//Proc of Global Telecommunications Conference. New York:[s.n.], 1997:1943-1947.

[6] 郭成城,晏蒲柳. 一种异构 Web 服务器集群动态负载均衡算法[J]. 计算机学报, 2005,28(2):179-184.

[7] 赵大勇. 业务选择网管中访问控制机制的研究[D]. 西安:西安电子科技大学,2008.

[8] 易勤勤. SSG 中多优先级队列管理机制的研究与设计[D]. 西安:西安电子科技大学,2008.

[9] CERPA A, ELSON J. The network element control protocol[EB/OL]. (2000-09-10)[2008-06-20]. http://www.circlemud.org/~jelson/writings/draft-cerpa-necp-03.txt.

(上接第 1678 页)

图4　音乐节拍实时跟踪演示系统

参考文献:

[1] OVERGOOR J. An evaluation method for audio beat detectors[C]//Proc of the 4th Twente Student Conference on IT. Enschede:[s.n.], 2006.

[2] MEUDIC B. A causal algorithm for beat-tracking[C]//Proc of the 2nd Conference on Understanding and Creating Music. 2002.

[3] DIXON S. Evaluation of the audio beat tracking system beat root[J]. Journal of New Music Research,2007,36(1):39-50.

[4] GOTO M. An audio-based real-time beat tracking system for music with or without drum-sounds[J]. Journal of New Music Research, 2001,30(2):159-171.

[5] Audio beat tracking[EB/OL]. (2007-08-02)[2008-03-20]. http://www.music-ir.org/mirex/2006/index.php/Audio_Beat_Tracking.

[6] DAVIES M E. Towards automatic rhythmic accompaniment[D]. Queen Mary: University of London, 2007.

[7] SCHEIRER E D. Tempo and beat analysis of acoustic musical signals[J]. Journal of Acoustical Society of America, 1998,103(1):588-601.

[8] STOWELL D, PLUMBLEY M D. Adaptive whitening for improved real-time audio onset detection[C]//Proc of International Computer Music Conference (ICMC'07). 2007.