第 34 卷 第 20 期 计 算 机 工 程 2008 年 10 月
Vol.34 No.20 Computer Engineering October 2008
·软件技术与数据库· 文章编号：1000—3428(2008)20—0069—02 文献标识码：A 中图分类号：TP301.6

哼唱检索中改进的动态时间规整算法

罗 凯，魏 维，谢青松
(西安通信学院一系，西安 710106)

摘 要：针对传统动态时间规整算法只考虑音高特征而不考虑音长特征的缺点，提出改进的算法，采用音高差和音长差共同构成算法中的代价函数，在此基础上实现了一个哼唱检索系统的原型。在数据库容量为 115 首乐曲和 118 个哼唱片段的测试中，该算法的前 10 位命中率为 81.0%，前 3 位命中率为 72.4%，性能优于 4 种同类算法。

关键词：哼唱检索；动态时间规整；旋律

Improved Dynamic Time Warping Algorithm in Query by Humming

LUO Kai, WEI Wei, XIE Qing-song
(The First Department, Xi'an Communication Institute, Xi'an 710106)

【Abstract】Aiming at the drawback that traditional Dynamic Time Warping(DTW) algorithm only considers pitch and hardly regards duration, this paper presents an improved algorithm with the help of note duration. Both delta pitch and delta duration are used to construct the cost function in the algorithm. A demo system of Query By Humming(QBH) is implemented. Experiments are conducted to test the algorithm with 118 humming clips and 115 songs in music database. The success rate of top-10 and top-3 of the improved algorithm are 81.0% and 72.4% respectively, exceeding the other four algorithms.

【Key words】Query By Humming(QBH); Dynamic Time Warping(DTW); melody

1 概述

哼唱检索(Query By Humming, QBH)是把用户哼唱的一个音乐片段作为系统的输入，通过检索音乐数据库，返回检索到的歌曲。由于用户哼唱时可能存在错误(如音符发音不准确)，检索结果往往是给出一张歌曲列表(如列出 10 首歌)，这张歌曲列表是根据歌曲间的相似度进行排序的，如果用户要检索的歌曲落于这张歌曲列表中，就认为这次检索正确命中。图 1 为哼唱检索系统的工作流程。

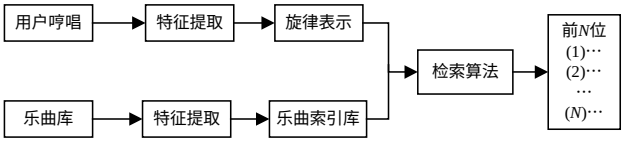

图 1 哼唱检索系统工作流程

文献[1]首次提出了 QBH 系统，该系统采用 3 步音高轮廓(分别用 U, D, S 表示后一个基音比前一个基音高、低或相等)方式，将旋律转化为以字符串表示，选择自相关函数法提取基音，使用近似字符串匹配算法计算相似度。为了有效描述旋律和改进检索方法，人们往往用音高和音长来描述旋律，并在此基础上提出了很多算法。如：基于音高轮廓几何相似匹配方法的大致思路是在二维空间中比较 2 条音高曲线的几何相似性；文献[2]将音高和节奏特征一并考虑，提出了线性对齐匹配法。虽然运用了音高和音长特征，但只是分别计算了两者各自的相似度，再将其合并加权。如何有效地利用音高和音长特征进行检索，也是人们进一步研究的方向。

本文提出了一种改进的基于音高和音长信息的动态时间规整(Dynamic Time Warping, DTW)算法，并在此基础上实现了一个哼唱检索系统的原型。

2 动态时间规整算法及其改进算法

检索算法是基于内容的音乐信息检索研究的核心，其性能直接影响检索结果和整个系统性能，既需要考虑节奏的变化、发音不准确、多音、漏音等，还必须考虑用户能够接受的平均检索时间等问题。本文的算法是在 DTW 算法的基础上引入音长信息，并充分结合音高和音长特征而设计的。

2.1 动态时间规整

动态时间规整算法是在一定的路径限制下寻找代价最小的最佳路径的方法，选择的路径方式与合理的代价函数具有重要的作用。在哼唱检索中，常用的路径方式有 2 种，如图 2、图 3 所示。本文采用了图 2 所示的路径方式。

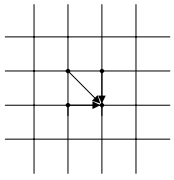

图 2 3 条路径的 DTW

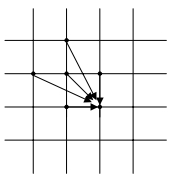

图 3 5 条路径的 DTW

设查询旋律为 $X=\{x_1,tx_1,x_2,tx_2,\cdots,x_m,tx_m\}$，对应的乐曲的旋律为 $Y=\{y_1,ty_1,y_2,ty_2,\cdots,y_n,ty_n\}$，其中，$x_i,y_j$ 为音高序列；tx_i,ty_j 为 x_i,y_j 对应的音长，则动态时间规整算法如下：

$$D(i,j)=\min\begin{cases}D(i-1,j)+d_{i-1,j}\\D(i-1,j-1)+d_{i-1,j-1}\\D(i,j-1)+d_{i,j-1}\end{cases} \quad (1)$$

作者简介：罗 凯(1982－)，男，硕士研究生，主研方向：音乐信息检索；魏 维，教授；谢青松，硕士

收稿日期：2007-12-10 **E-mail：**luokai-xy-jx@163.com

其中，D 是 2 个查询片段之间的距离；d 是代价函数，具体计算如下：

$$d_{i-1,j}=|x_i|+c \tag{2}$$

$$d_{i-1,j-1}=|x_i-y_j| \tag{3}$$

$$d_{i,j-1}=|y_j|+c \tag{4}$$

其中，c 为平衡因子，目的是平衡插入或删除音符所带来的代价。

2.2 改进的动态时间规整算法

音长的引入可以提高检索效果，问题是如何有效地引入音长。2.1 节的 DTW 算法只用到了音高差信息，没有考虑音符的音长信息。如图 4 所示，左边和右边所表达的旋律是截然不同的，而这在动态时间规整算法中却没有区别。

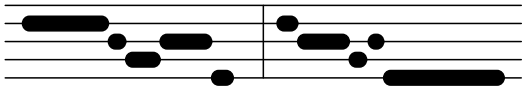

图 4 乐曲库中的旋律和哼唱旋律[3]

如果一个音符的最短时间为 100 ms，当哼唱信号分帧的帧长为 20 ms、帧移为 10 ms 时，一个音符的音长会有几十帧甚至上百帧，那么 2 个音符的音长差可能也会有几十帧甚至上百帧，而音高差的变化主要集中在[-4, 4]。与音高差的处理方法类似，本文也以音长差替代绝对音长。为了将音长差引入 DTW 算法中，必须对其作如下处理：

(1)算出音高差序列的平均值：

$$\bar{P}=\frac{1}{N}\times(\sum_{i=1}^{N}|x_i|) \tag{5}$$

(2)算出音长差序列的平均值：

$$\bar{T}=\frac{1}{N}\times(\sum_{i=1}^{N}tx_i) \tag{6}$$

(3)得到变换后的音长差序列：

$$tx_i=\bar{P}\times\frac{tx_i}{\bar{T}} \tag{7}$$

变换后的音长差序列和音高差序列共同构成代价函数：

$$d_{i-1,j}=(1+|x_i|)(1+\frac{|tx_i|}{k})+c \tag{8}$$

$$d_{i,j}=(1+|x_i-y_j|)(1+\frac{|tx_i-ty_j|}{k}) \tag{9}$$

$$d_{i-1,j}=(1+|y_i|)(1+\frac{|ty_i|}{k})+c \tag{10}$$

其中，k 是音高差和音长差之比。式(2)~式(4)中的 $|x|$ 相应地改为式(8)~式(10)中的 $1+|x|$。用 $1+|t|/k$ 表示引入的音长差。

3 旋律提取

旋律提取是从哼唱信号中提取音高或音长等信息，用于表示旋律。国外有文献对音乐记忆特性进行了研究，指出旋律的轮廓比精确的旋律更易于记忆，于是，人们采用了各种方法描述旋律。文献[4]采用了三元组(音高轮廓，音程，音长)表示旋律。文献[2]用(音差，时间)描述每个音符，其中，音差指与上一个音符频率的差值；时间指该音符开始的时刻。文献[5]使用相邻音符的音高差和音长之比表示旋律。本文从时域中分析哼唱信号，采用三元组(音高差，音符起始时间，音符结束时间)表示旋律。

哼唱信号是在安静的环境下录制的，且限制用户只能以“Da…或 La…”等形式哼唱，录制格式为 22 050 Hz/16 bit/双声道。这些数据并不适合直接提取旋律，而应做预处理：(1)选取声道。录制信号为双声道，从中取一个声道即可。(2)将信号降采样到 8 kHz。(3)将哼唱数据进行分帧，帧长为 20 ms，帧移为 10 ms。旋律提取过程如图 5 所示。

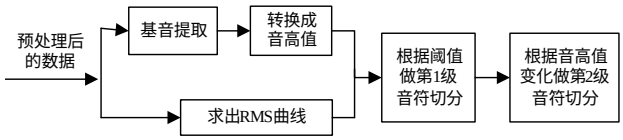

图 5 旋律提取过程

3.1 基音提取

本文采用自相关函数法提取基音，并将哼唱声音的基频范围限定为[100 Hz, 500 Hz]。提取经预处理后的每帧信号的基音，如果提取的基音超出范围，则将这帧信号的基音值定为 0。根据式(11)，将各帧信号的基音转化为标准的 MIDI 音高值：

$$f_{\text{Semitone}}=12\times\text{lb}(f_{\text{Hz}}/440)+69 \tag{11}$$

3.2 音符切分

本文采用了 2 级切分音符的方法，先根据能量做第 1 级切分，将停顿较为明显的音符切分出来，再对连续的发音段采用音高变化做第 2 级切分，最后检测音符的有效性，并根据规则合并、删除音符。

由能量切分可以得到音符的发音段，设定一个音符的最短时间为 100 ms，当发音段长度小于 100 ms 时，认为这个发音段无效；当发音段长度小于 200 ms 时，认为只有一个音符，不做第 2 级切分，取得发音段内平均音高值为这个音符的音高值；当发音段长度超过 200 ms 时，进行第 2 级切分，将音高变化明显的发音段切分出来，最后确定音符的有效性。

4 性能测试

为了测试改进算法的性能，引入了 N-gram 法，近似字符串匹配算法 1(只考虑音高差)，近似字符串匹配算法 2(考虑音高差和音长差)和动态时间规整算法进行比较。为了便于书写，将 5 种算法分别标记为 DP(近似字符串匹配 1)、DP2(近似字符串匹配 2)、N-gram、DTW(动态时间规整)和 DTW2(改进的动态时间规整)。

在 DP 算法中，相似度是根据编辑距离排序的，详细介绍可以参考文献[1,6]；在 DP2 算法中，由音高和音长计算得到的相似度是在规划矩阵级别上进行合并处理的，而不是简单的加权合并，其具体过程可以参考文献[6]；在 N-gram 算法中，将长度为 3 的全部音程序列片段索引出来，其相似度的计算可以参考文献[7]；DTW 算法中的路径方式和 DTW2 相同，相似度是根据 DTW 距离排序的，其中，c =1，详细内容可以参考文献[7]；在 DTW2 算法中，k =10, c =3。

由 6 位测试人员(3 男 3 女)录制了 118 个哼唱片段，测试者均无音乐专业背景，每个哼唱片段长度为 5 s~10 s。数据库容量为 115 首 MIDI 乐曲。各算法前 10 位命中率如图 6 所示。可以看出：(1)DP,DP2 和 N-gram 的性能比较差，前 10 位命中率分别为 37.1%, 44.8%和 44.0%。主要原因是 DP 和 DP2 用 3 个字符表示旋律，使得乐曲间的差异性非常小，影响了检索性能，而 N-gram 是一种基于统计的算法，没有考虑音符之间的时序关系。(2)DTW 和 DTW2 的性能相对于前 3 种算法有了明显的提高，前 10 位命中率分别为 78.5%和 81.0%。(3)加入音长差的 DP2 和 DTW2 均比没有加入音长差的 DP 和 DTW 的性能好。因此，音长是音符的重要信息，合理地利用音长信息将有助于提高检索性能。 (下转第 73 页)