ISSN 1000-0054 清华大学学报（自然科学版）2009 年 第 49 卷 第 S1 期 28/31
CN 11-2223/N J Tsinghua Univ (Sci & Tech), 2009, Vol. 49, No. S1 1402-1407

用于哼唱识别精确匹配的线性伸缩动态规划算法

曹文晓[1,2]， 刘 轶[1,2]， 郑 方[1,2]， 蒋丹宁[3]， 秦 勇[3]

（1. 清华大学 信息技术研究院，语音和语言技术中心，北京 100084；2. 清华信息科学技术国家实验室 技术创新与开发部，语音和语言技术中心，北京 100084；3. IBM 中国研究中心，北京 100094）

摘 要：提出一种用于哼唱识别精确匹配的线性伸缩动态规划算法。该算法将哼唱旋律切割成句子，对每一句子进行线性伸缩匹配，同时在句子层次进行动态规划获得最优路径。该算法更有效地利用了哼唱语音的分段特性并克服了动态规划在长路径搜索时可能丢失全局最优路径的缺点。在含5 223 首MIDI的数据库上同等条件下该算法正确率分别比线性伸缩、动态规划及递归匹配方法提高10.5%、6.0%和2.8%。该算法具有更高的准确率和更小的时间复杂度，是一种更有效的精确匹配算法。

关键词：检索机；哼唱识别；精确匹配；线性伸缩；动态规划

中图分类号：TP 391.3 **文献标识码**：A

文章编号：1000-0054(2009)S1-1402-06

Linear scaling based dynamic programming algorithm for accurate matching in QBH

CAO Wenxiao[1,2], LIU Yi[1,2], ZHENG Fang[1,2], JIANG Danning[3], QIN Yong[3]

(1. Center for Speech and Language Technologies, Research Institute of Information Technology, Tsinghua University, Beijing 100084, China;
2. Division of Technology Innovation and Development, Center for Speech and Language Technologies, Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China;
3. IBM China Research Lab, Beijing 100094, China)

Abstract: A linear scaling (LS) based dynamic programming (DP) algorithm was developed for accurate matching of queries by humming. The query contours are split into phrases, with the LS match calculated for each phrase. Finally dynamic programming is used to analyze on all the phrases to choose the optimal matching path. The algorithm more efficiently considers the query contours related to the phrases, thus, overcoming the missing-global-optimal-path disadvantage of dynamic programming for long path matching. Tests on a 5 223 MIDI database show that the algorithm outperforms the traditional LS method by 10.5%, the DP method by 6.0% and recursive alignment by 2.8% for the top-1 rate. Thus, the algorithm is more efficient and more accurate while being less expense.

Key words: search engine; query by humming; accurate match; linear scaling; dynamic programming

随着信息技术的发展，多媒体搜索技术的应用越来越广泛。哼唱识别(query by humming, QBH)是音乐搜索中的一个重要应用，它使用户能通过哼唱一段旋律来检索歌曲。哼唱识别与传统音乐搜索引擎不同，传统音乐搜索引擎基于音乐的描述信息如歌手、歌名等信息进行搜索，而哼唱识别基于音乐内容。用户喜欢一首歌，通常喜欢该歌曲的某一段旋律，对该段旋律的记忆更为深刻。哼唱识别使用户不必记忆歌曲的描述信息，直接哼唱最喜欢的旋律搜索，使音乐搜索更为自然、简单且人性化。哼唱识别技术的发展将给音乐搜索带来更广阔的应用前景。

近年来哼唱识别研究有很大进展。哼唱识别技

收稿日期：2009-03-13
基金项目：IBM 与清华大学合作项目 (2007-2008)
作者简介：曹文晓(1984—)，男(汉)，湖南，硕士研究生。
通讯联系人：刘轶，副研究员，E-mail: eeyliu@tsinghua.edu.cn

术的核心是用户哼唱旋律与数据库中的旋律间的匹配算法。识别技术应用的关键是响应速度和准确率, 分别取决于快速匹配和精确匹配[1-2]。本文侧重研究后者。文[3]将音乐作为时间序列并在DTW 基础上进行一种称为“Envelope Transforms”的变换来计算时间序列间的距离, 相比DTW 大大缩短了计算时间同时又保证了准确率。文[4]则使用动态规划(dynamic programming, DP)和启发式搜索分别解决哼唱旋律和模板旋律间在速度和音高上的变化问题, 相比单纯的动态规划获得了更高的准确率同时提高了响应速度。文[1]在线性伸缩基础上使用一种自顶向下递归匹配(recursive alignment, RA)的方法获得了比线性伸缩及动态规划更好的效果。文[5]基于Markov 模型进行哼唱识别并证明哼唱时间越长识别效果越好。

上述这些精确匹配方法有一些获得了很好的效果, 但这些方法也忽略了哼唱识别的某些特点。首先, 哼唱的句子间存在较明显的停顿, 因而对于哼唱旋律, 可以划分出不同的句子。针对哼唱的不同句子分别进行匹配将获得更好的效果; 其次, 用户的哼唱输入起始点一般是歌曲句子的起始点, 因而对模板旋律作句子起始点标记可以有效地减小匹配的计算量; 再者, 哼唱的音高和时长相比模板旋律都会有一定偏移, 在计算二者间的距离前, 需要估计速度变化及音高偏移的大小并调整使得二者速度和音高在同一基准线上。

本文提出的线性伸缩动态规划算法正是在上述分析下提出的。线性伸缩动态规划是一种对哼唱旋律进行句子切分后分段进行线性伸缩匹配然后在句子层次上进行动态规划匹配的方法。通过检测指定长度的静音将哼唱旋律切分成不同句子, 然后对每一句子根据它们的时间长及起始点和数据库中的模板旋律进行线性伸缩匹配, 每个句子通过调整其结束点位置可有多种匹配方案, 同时使用动态规划获得整体最优匹配分数。这种方法有效利用了哼唱由多个句子组成并具有明显界限这一特点, 以句子为单位进行动态规划, 有效克服了动态规划长距离时可能丢失全局最优路径的缺点[1], 同时不同句子使用不同的线性伸缩参数也更具合理性。

1 哼唱识别系统整体框架描述

本文的哼唱识别系统整体框架如图1 所示。系统由3 个模块构成。系统将用户的哼唱输入经特征提取后与旋律数据库中的模板旋律进行快速过滤及精确匹配, 最后给出候选的歌曲。

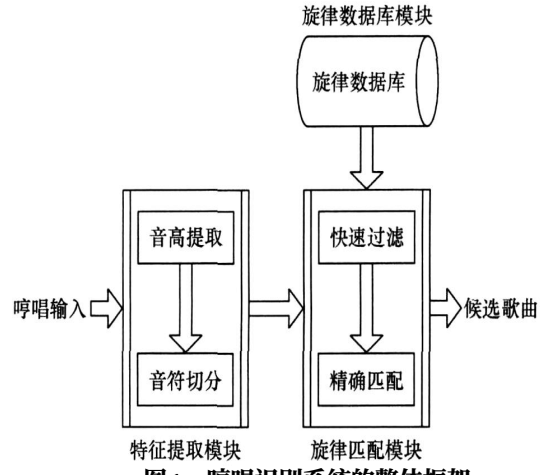

图1 哼唱识别系统的整体框架

特征提取模块是系统前端, 包括音高提取和音符切分。音高提取将输入语音按帧切分后提取每帧的频率信息并转换成音高信息, 得到音高序列。音符切分过程将音高序列切分成不同音符。

旋律数据库模块用于管理并索引MIDI的模板旋律信息。负责从MIDI的多轨旋律中找出歌曲主旋律, 同时记录歌曲中每个句子的起始点位置。由于哼唱通常从句子最前面开始, 标记句子起始点位置有利于降低旋律匹配的搜索空间并提高系统性能。

旋律匹配模块包括快速过滤及精确匹配过程, 快速过滤的主要目的是以最短的时间去除大部分不可能的候选, 可以由多层过滤器组成。精确匹配则对快速过滤后的候选进行更精确的计算和筛选, 最终给出前N 个候选。快速匹配和精确匹配构成系统响应速度和准确率的平衡。

2 哼唱识别中的精确匹配方法

2.1 精确匹配

1) 精确匹配的定义。

本文使用的模板旋律为MIDI格式。用户哼唱的语音被按帧切分后提取成音高序列, 表示成$Q=\{q_1, q_2, \cdots, q_m\}$, 其中$m$ 是哼唱语音帧数, q_i 表示第i 帧的语音对应的音高。和Q 匹配的MIDI旋律表示成$P=\{(p_1, d_1), (p_2, d_2), \cdots, (p_n, d_n)\}$, 这里$n$ 表示MIDI旋律包含的音符数, p_i 表示第i 个音符的音高, d_i 是第i 个音符的时长, 用帧数表示。上述n 是一个估计的值, 估计方法参考2.2节。定义距离函数$\mathrm{dist}(q_i, p_j)$ 表示哼唱旋律中的音高q_i 和MIDI旋律中的音符音高p_j 间的距离。由此, 精确匹配定义为: 给定用户查询Q 及MIDI片段P, 以及音高间的距离函数$\mathrm{dist}(q_i, p_j)$, 如何计算Q 和P 的最小距离

FineDist(Q, P, dist)。

2) 精确匹配的常用方法。

线性伸缩(linear scaling, LS)是一种简单快速而有效的方法,这种方法通过线性地拉伸或收缩一段旋律来和另一段旋律进行匹配。这种方法被证明比经过良好训练的HMM 模型具有更好的效果[6],这也说明了哼唱和MIDI旋律之间确实存在线性的关系。这种方法的算法描述如下。

算法 1 LS(Q,P):

1) INPUT: $Q=(q_1,q_2,\cdots,q_m)$
$P=((p_1,d_1),(p_2,d_2),\cdots,(p_n,d_n))$

2) $j\leftarrow 1$

3) melodyDur$\leftarrow\sum_{k=1}^{n} d_k$, $D\leftarrow 0$,
scale$\leftarrow n/$melodyDur

4) for $i=1\rightarrow m$do

5) $s\leftarrow\lfloor j+\text{scale}\times d_i\rfloor$

6) $D\leftarrow D+\sum_{k=j}^{s}\text{dist}(q_k,p_j)$

7) $j\leftarrow s+1$

8) end for

9) return($-D$)

这种算法将哼唱旋律和MIDI旋律整体进行线性伸缩匹配,但实际上哼唱旋律和MIDI旋律之间并不是严格的满足线性关系,特别是句子数越多,时间越长,就越不能保证两个旋律之间的线性关系,尽管局部可以满足线性关系。因此这种方法尽管有效却很难获得更高的准确率。

另一种方法是动态规划。动态规划方法比线性伸缩具备更多的灵活性,具有更大的搜索空间,因而这种方法能够获得比线性伸缩更高的准确率,但时间复杂度也比线性伸缩高。动态规划算法一般基于帧进行匹配,在这种条件下,MIDI旋律P 的音符序列被打散成以帧为单位的音高序列$\{p_1', p_2', \cdots, p_t'\}$,假设$D_{ij}$表示动态规划匹配到哼唱旋律$Q$ 的第i 帧和MIDI旋律P 的第j 帧时的最优分数。则动态规划算法中存在如下的递归式:

$$D_{ij}=\text{dist}(q_i,p_j')+\min(D_{(i-1)j},D_{(i-1)(j-1)},D_{i(j-1)}). \tag{1}$$

动态规划的方法相比线性伸缩能获得更好的对齐结果,但是其时间复杂度也比线性伸缩高,因此为了提高准确率并降低时间复杂度,就产生了动态规划的各种变形,它们被证明比传统动态规划具有更好的效果[4,7]。如图2 是本文实现的一种路径受限的动态规划的演示图,这种方法限制动态规划的路径必须在$y=\alpha x$ 及$y=\beta x$ 两条直线之间,其中$0<\alpha<1$, $\beta>1$, 另外在每一步匹配后按照一个给定的剪枝参数beam 进行剪枝,加快速度。

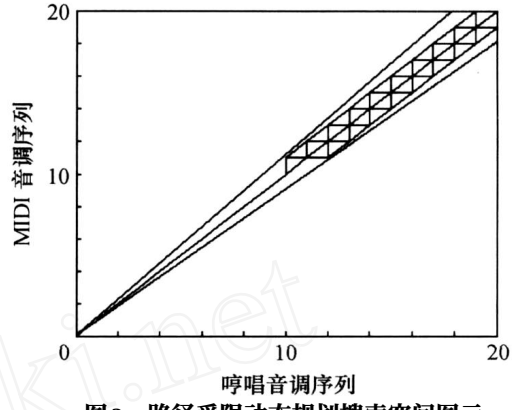

图2 路径受限动态规划搜索空间图示

RA 的方法通过二分法多层次地分段进行线性伸缩匹配,达到更好的效果。RA 的方法来源于线性伸缩但克服了线性伸缩的弱点。它自顶向下每次取MIDI旋律的中点,然后在哼唱旋律中找到最佳的对齐点,将MIDI旋律和哼唱旋律同时分成两段后继续进行递归的匹配,如此进行多层次的匹配,直到达到终止条件。RA 的层次越多,匹配越细,时间复杂度越高。这种方法相比动态规划能够更好地匹配音乐旋律的总体轮廓特征,更容易找到全局的最优路径。但是以哼唱为整体进行切分,没有区分句子的概念,切分的片段和MIDI之间不一定满足线性关系,因而以中点切割后分段使用不同的线性伸缩系数也还是有一定的缺陷。

2.2 速度变化及音高偏移

在实际的哼唱识别中,哼唱旋律与MIDI旋律间存在速度变化和音高偏移,即哼唱输入一般与MIDI库中的旋律存在差异。因此,在已知MIDI旋律起始点的条件下(本文标注了MIDI中句子之间的切分点),进行精确匹配前,必须先估计哼唱旋律相对MIDI旋律的速度比以及音高偏移,找出最佳的速度比及音高偏移。确定了最佳速度比及音高偏移之后,也就可以估计2.1中的n 值了。

最佳速度比及音高偏移的估计参考了文[4]使用启发式的算法,假设哼唱旋律为$Q=\{q_1, q_2, \cdots, q_m\}$, 候选MIDI旋律为$P=\{(p_1,d_1),(p_2,d_2),\cdots\}$, 最佳速度比为$R_b$, 最佳音高偏移为$S_b$, 则估计最佳速度比及最佳音高偏移的算法如下。

输入: 速度比区间[0.5, 2]的划分点数l_1, 音高偏移的估计区间长度l_2, l_1、l_2为奇数。速度比初始化值$r=1$ 音高偏移初始化值$s=q_1-p_1$。

1) 初始化。

$c_1 \leftarrow \lfloor l_1/2 \rfloor$, idx$=c_1$

rbin$[c_1] \leftarrow r$

rbin$[i] \leftarrow 0.5+i(r-0.5)/c_1$, $0 \leqslant i < c_1$

rbin$[j] \leftarrow r+j(2-r)/c_1$, $c_1+1 \leqslant j < l_1$

2) LS 计算距离。

$$\begin{cases} \text{left} \leftarrow LS'(Q,P,\text{rbin}[\text{idx}-1],s) \\ \text{current} \leftarrow LS'(Q,P,\text{rbin}[\text{idx}],s) \\ \text{right} \leftarrow LS'(Q,P,\text{rbin}[\text{idx}+1],s) \end{cases}$$

3) 找出最小值并更新 idx。

若 left$=$min(left, current, right)则:

newidx$\leftarrow$idx-1

否则若 right$=$min(left, current, right)则:

newidx$\leftarrow$idx$+1$

否则

newidx$\leftarrow$idx

4) 检查结束条件。

若 newidx$=$idx 或 newidx<0 或 newidx$\geqslant l_1$ 则:

$R_b \leftarrow$rbin[idx]并跳出循环

否则

idx$\leftarrow$newidx 转步骤 2。

上述过程中的$LS'(Q,P,r,s)$表示用速度比r及音高偏移s对 MIDI 旋律P进行调整后计算Q、P的 LS 距离, LS 距离的计算方法参考 2.1 节。音高偏移的最佳值估计和速度比类似,唯一的区别就是初始化过程变成:

$c_2=\lfloor l_2/2 \rfloor$, idx$=c_2$,

sbin$[c_2]=s$,

sbin$[i]=(s-c_2)+i$, $0 \leqslant i < c_2$,

sbin$[j]=s+(j-c_2)$, $c_2+1 \leqslant j < l_2$.

同时$LS'(Q,P,r,s)$函数中使用R_b作为速度比, 最后估计出最佳音高偏移S_b。

上述的精确匹配定义是在速度和音高已经经过调整的条件下给出的。

3 线性伸缩动态规划算法

3.1 原 理

本文提出了一种线性伸缩动态规划(linear scaling based dynamic programming, DPLS)算法。该算法的基本思想是:

1) 哼唱旋律可以根据停顿切分成多个句子, 每个句子与MIDI旋律之间满足线性关系, 使用线性伸缩匹配;

2) 单个句子的切分不做严格限制, 可调整句子结束点, 得到多个候选句子, 进行线性伸缩匹配;

3) 句子层次上, 使用动态规划, 选择最好的切分结果, 取该切分的匹配分数作为最佳匹配得分。

在实际的QBH 系统中, 用户的哼唱在不同句子之间一般是存在明显的停顿的, 这是由于唱歌时换气造成的。这个停顿的时间一般比句子中间的停顿要长, 因而用户的哼唱可以切分成不同句子。另一方面, 在用户不存在严重跑调的情况下, 句子和对应的MIDI旋律之间存在线性的变化, 如速度快或速度慢, 因而在句子层次上进行线性伸缩匹配是合理的; 再者, 由于在句子之间过渡时可能存在一些不合理的音符如用户换气引入的喘气声, 可以对句子的结束点位置进行微调, 取最好的匹配结果; 最后, 以帧为单位的动态规划存在路径长、可能丢失全局最优路径的缺点, 而在句子的层次上动态规划的路径长度大大缩小, 因而避免了动态规划在长路径时可能丢失全局最优路径的缺点。

从而, 本文提出的算法是在局部使用线性伸缩匹配的条件下在全局获取最优匹配路径。

3.2 算法设计及实现

在2.2中已经给出速度比及音调偏移的估计算法, 这里假设已经估计得到哼唱和MIDI旋律的速度比及音调偏移, 则在MIDI旋律和哼唱旋律已调整到速度和音调一致的条件下。算法的描述如下。

令$Q=\{q_1, q_2, \cdots, q_m\}$表示哼唱的音高序列, q_i表示第i帧的音高, m为哼唱的帧数。

令$P_s=\{(p_s, d_s), (p_{s+1}, d_{s+1}), \cdots\}$表示要匹配的MIDI音符序列, 该MIDI音符序列的起点为第s个音符, 长度不定, 最多至MIDI结束。

令$P_{ij}=\{(p_i, d_i), (p_{i+1}, d_{i+1}), \cdots, (p_j, d_j)\}$表示MIDI旋律$P_s$中从$i$到$j$的音符构成的子序列。

1) 将哼唱音高序列根据指定长度静音切分成句子序列$\{Q_1, Q_2, \cdots, Q_h\}$, 其中$h$表示哼唱音高序列切分得到的句子数, 句子$Q_i=\{q_{s_i}, q_{s_i+1}, \cdots, q_{e_i}\}$, 其中$s_i$、$e_i$分别表示第$i$个句子的起始和结束帧。

2) 令 tokens$_k$ $(0 \leq k \leq h)$表示匹配到第k个句子时的所有可能的token, token 表示当前已匹配的所有句子和与它们匹配的MIDI音符序列的一种对齐结果, 每一个token 包含当前已匹配到的MIDI旋律

的结束音符下标endnote及累积匹配得分score。

初始化第一个token:

token.score= 0, token.endnote= 0

$tokens_0$= {token}

3) 对句子序列$\{Q_1, Q_2, \cdots, Q_h\}$中每一$Q_k$:

a) 将Q与P_s的起始时间对齐,获取Q中句子Q_k在P_s中对应的结束音符位置E_k;

b) 对$tokens_{k-1}$中的每个$token_i$:

对所有$e= E_k- t, \cdots, E_k+ t$:

令 melodyStart= token$_i$.endnote,
melodyEnd= e,

$token_i'$ = token$_i$.score + LS(Q_k, $P_{melodyStart, melodyEnd}$) $token_i'$.endnote= e

将$token_i'$加入$tokens_k$。

上述的t为预先设定的参数,$t \in Z$, $t \geq 0$, LS(Q_k, $P_{melodyStart, melodyEnd}$)是线性伸缩匹配函数,2.1中已给出算法。

c) 对$tokens_k$中的所有token进行归并,结束音符下标相同的保留匹配得分小的,然后对剩余token排序,保留前beam个得分最小的token,beam为预设参数。

4) 对$tokens_h$中所有token按照分数排序,假设最低分为score,取-score/m作为匹配分数。

这种算法的特点是在哼唱旋律和MIDI旋律匹配的过程中在整体进行速度和音高的同步,在哼唱的不同句子匹配时也使用特定的拉伸系数,同时在句子层次上使用动态规划,因此可以看成在线性伸缩基础上的动态规划。其优点是在将线性伸缩的参数细化的同时简化了动态规划过程,克服了动态规划长距离时可能丢失全局最优路径的缺点。

图3给出了该算法的一个示例。哼唱音高序列共有62帧被切分成3个句子,然后逐个与MIDI音高序列进行匹配,在不剪枝的条件下,图中显示了所有可能的路径,这里的参数t= 1。可见本算法相比动态规划缩小了搜索空间。

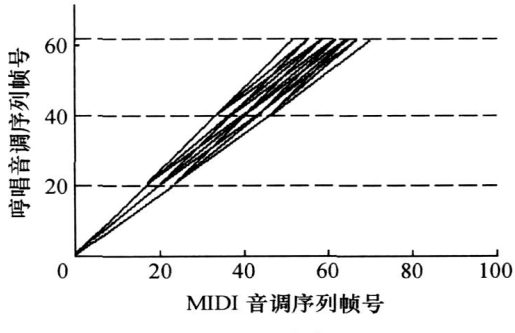

图3 DPLS算法搜索空间示例

3.3 DPLS与其他算法的对比分析

这里仅将DPLS算法与2.1中的所述算法进行比较,给出各算法的特性对比如表1所示。表中的线性伸缩是指算法是否利用了线性伸缩的特性。分段匹配是指算法是否对旋律进行分段后再分别匹配。分段界限是指分段的依据。全局优化是指搜索最优路径。由对比结果可以看出,DPLS具有比其他算法更多的优点,DPLS能够获得更高的准确率。

表1 DPLS与其他算法的特性比较

特性	LS	DP	RA	DPLS
线性伸缩	满足	不满足	满足	满足
分段匹配	无	无	有	有
分段界限	无	无	中点	句间停顿
全局优化	无	有	有	有

4 实验及结果

4.1 系统描述

系统的框架在第1节中已经描述,见图1。哼唱语音以8 kHz频率采样,首先被转换成音高序列,窗宽为100,帧移为80,每秒为100帧,然后音高序列被切分成音符序列。数据库中的每个MIDI旋律被切分成多个候选,每个句子起始点之后的旋律作为一个候选。旋律匹配模块中的快速过滤使用LS过滤器,即哼唱旋律与所有候选旋律进行LS匹配后,将分数排序后输出前500个候选作为精确匹配的输入。同时在快速过滤过程中,对每个候选估计和哼唱旋律间最佳的速度比和音调偏移,用于精确匹配前对旋律的调整同步,估计的算法见2.2。精确匹配使用各种精确匹配算法,给出前20的候选。

4.2 实验数据

本文使用的哼唱测试数据集共含355个哼唱wav文件,每个wav文件以8 kHz采样率16 bit编码录制,总长度约71 min。在这些哼唱输入中,多数都是带歌词哼唱的。这些测试数据是2008 MIREX QBH evaluation的一个标准测试集(task2),由中国科学院声学所中科信利实验室提供。

MIDI数据库的MIDI被作者自己开发的标注工具手工标注了句子起始点。MIDI数据库共包含5 223首歌曲,其中有106首目标MIDI,其他歌曲作为噪声数据。106首目标歌曲被我们开发的标注工具标注成了2 213段。上述数据库中目标歌曲和噪声歌曲之间没有重叠。

4.3 实验结果

为了证明本文提出的DPLS算法的有效性, 本文实现了几种已有的方法作为对比。由于测试数据、数据库、测试环境等方面的差异, 不同算法之间很难做绝对的比较。因此定义了包括TOP1、MRR以及TME的衡量标准。TOP1即正确率, 为最佳候选正确的比率。TME为响应时间。MRR定义如下。

假设第i个哼唱的目标MIDI在候选结果中的排名为R_i, 则:

$$MRR = \sum_{i=1}^{n} \frac{1}{R_i} I_{R_i \le 20}; \tag{2}$$

$$I_{R_i \le 20} = \begin{cases} 1, & R_i \le 20; \\ 0, & \text{其他} \end{cases} \tag{3}$$

本文对2.1中提到的线性伸缩、路径受限的动态规划、递归匹配以及本文提出的线性伸缩动态规划算法进行了比较。这些比较实验中, QBH系统中除了精确匹配其他实验条件保持一致。各算法的基本描述如表2所示。

表2 实验比较的各算法描述

算法	描述
LS	算法见2.1
DP	路径受限的DP算法, 见2.1, 已优化参数, 最优参数为: beam=3, α=0.88, β=1.12
RA	文[1], 已优化参数, 递归层次最优为R=4
DPLS	本文提出算法, 已优化参数, t=2, beam=5

从表3的实验结果可知, 4种算法的性能比较结果为: LS<DP<RA<DPLS。LS为DP的特殊情况(即beam=1, $\alpha \to 1$, $\beta \to 1$), 搜索空间小于DP, 故DP优于LS。而RA更易于从全局找到最优结果, 克服了DP在路径过长时可能丢失全局最优路径的弱点, 故RA优于DP, 而DPLS将DP的方法上升到句子层面, 大大缩短了匹配距离, 克服了DP长距离时可能丢失全局最优的缺点, 同时DPLS又将分段匹配划分到句子层次, 相比RA的分段方法更具合理性, 因而DPLS优于RA。表3的结果也支持了本文前述对DPLS的分析。DPLS的准确率高于其他3种算法, 而响应时间仅高于LS算法。

表3 各算法在5 223首MIDI数据库上的性能比较

算法	TOP1/%	MRR	TME/s
LS	57.46	0.607 9	26.00
DP	58.72	0.621 3	28.41
RA	60.56	0.640 0	27.36
DPLS	62.25	0.653 3	26.67

5 结 论

本文提出一种用于哼唱识别精确匹配的线性伸缩匹配动态规划算法。该算法将哼唱旋律分割成句子, 以句子为单位进行线性伸缩匹配, 同时在句子的基础上通过动态规划获得最优路径。相比其他方法, 这种方法利用句子分界点进行更合理的分段线性伸缩匹配, 同时将动态规划提升到句子层次上, 缩短了动态规划的路径长度, 有效避免了动态规划在路径过长时容易丢失全局最优路径的问题。

本算法在含5 223首MIDI的数据库上和其他算法即LS、DP、RA进行比较, 比较结果证明了本算法的有效性。实验结果表明DPLS的准确率高于其他3种算法, 在5 223首MIDI的数据库上DPLS的准确率相比LS、DP、RA分别提高10.5%、6.0%和2.8%, MRR相比LS、DP、RA分别提高7.5%、5.2%和2.1%。因此, DPLS是一种更有效的精确匹配方法。

参考文献 (References)

[1] WU Xiao, LI Ming, LIU Jian, et al. A top-down approach to melody match in pitch contour for query by humming [C]//ISCSLP2006. 2006.

[2] Bainbridge D, Dewsnip M, Witten I. Searching digital music libraries. Digital libraries: People, knowledge, and technology [C]//5th International Conference on Asian Digital Libraries. 2002.

[3] ZHU Yunyue, Shasha D. Warping indexes with envelope transforms for query by humming [C]//Proc ACM SIGMOD Int Conf on Management of Data. 2003.

[4] Jyh-Shing Roger Jang, Gao Ming Yang. A query-by-singing system based on dynamic programming [C]//International Workshop on Intelligent Systems Resolutions (the 8th Bellman Continuum). 2000.

[5] Shifrin J, Birmingham W. Effectiveness of HMM-based retrieval on large databases [C]//Proc of International Symposium of Music Information Retrieval (ISMIR). 2003.

[6] Chao-Ling Hsu, Jyh-Shing Roger Jang. Continuous hmm and its enhancement for singing/humming query retrieval [C]// Proc of ISMIR. 2005.

[7] Mazzoni D, Dannenberg R B. Melody matching directly from audio [C]//ISMIR 2nd Annual International Symposium on Music Information Retrieval. 2001.