

基于语句特征的音乐哼唱快速检索技术

王培培¹, 杨晓春¹, 王 斌¹, 王晓晔²
(1. 东北大学 计算机科学与工程学院, 辽宁 沈阳 110169; 2. 中国人民解放军95806部队, 北京 100076)

摘 要: 哼唱检索作为音乐检索的重要方式, 由于其有效性和方便性, 引起了广泛的关注. 本文提出了一种新的基于语句特征的音乐哼唱快速检索技术, 可以实现哼唱音乐的快速检索. 该技术将音乐数据库和用户提供的哼唱片段, 按自然停顿方式划分音乐语句, 使用 BDTW 算法对音乐语句片段进行音高相似性计算, 并允许用户根据自己哼唱情况, 对匹配条件进行个性化设置, 限制数据库音乐片段和查询序列的局部最大差异长度. 另外, 对音乐库建立支持音乐语句查询的索引结构 DIS, 减少了检索时间. 实验结果表明所提出的检索方法能够快速有效地返回查询结果.

关 键 词: 音乐检索; 哼唱检索; 全序列匹配; 子序列匹配; DTW 算法

中图分类号: TP 31 文献标志码: A 文章编号: 1005-3026(2017)03-0315-05

Rapid Retrieval Technology of Query by Humming Based on Sentence Features

WANG Pei-pei¹, YANG Xiao-chun¹, WANG Bin¹, WANG Xiao-ye²
(1. School of Computer Science & Engineering, Northeastern University, Shenyang 110169, China; 2. China People's Liberation Army Troops 95806, Beijing 100076, China. Corresponding author: WANG Pei-pei, E-mail: wangpeipei@research.neu.edu.cn)

Abstract: As an important way of music retrieval, query by humming has gained wide attention because of its effectiveness and convenience. A novel retrieval technology of humming was proposed based on sentence features, which could provide fast retrieval for query by humming. In the proposed technology, the music database and humming given by users were first partitioned according to natural pauses, and then the BDTW (bounded dynamic time warping) algorithm was adopted to compute pitch similarity. In addition, users can also establish personalized settings in accordance with their own humming, and limit the maximum local length variance between music database fragments and query sequences. In addition, the index structure DIS was established to support music sentence query, which could reduce searching time. The experimental results verified both the efficiency and effectiveness of the proposed retrieval method.

Key words: music retrieval; query by humming; whole matching; subsequence matching; dynamic time warping (DTW) algorithm

随着网络技术的迅速发展, 现代多媒体信息的数据量正在急剧增多. 音乐作为最常见的声音媒体, 自动、准确、快速地查找音乐目标, 是一个既迫切又具有挑战性的研究课题. 哼唱查询 (query by humming) 是音乐查询中一项重要内容, 即根据哼唱片段搜索音乐数据库的最相似的 k 个歌曲.

1 相关工作

哼唱检索技术起步比较晚, 1995 年 Ghias 等^[1]对音乐哼唱检索进行了开创性研究, 使用近似字符串匹配算法实现音乐检索. 2003 年, Zhu 等^[2]使用动态时间规整 (dynamic time warping,

DTW)技术,将哼唱片段直接与数据库中的歌曲进行全序列匹配.2006 年,Unal 等^[3]将 N-gram 反向索引结构用于主题挖掘的音乐数据,提高了检索效率.2011 年,Kotsifakos 等^[4]使用子序列匹配方式,可以解决哼唱片段中一些音符或音节丢失的问题,并在 2012 年给出了该方法的系统^[5]实现.此外,近年来字符串匹配技术^[6-7]的快速发展,对音乐检索有很好指导作用,2016 年,Wang 等^[8]提出多维音乐序列匹配技术.

2 基于语句特征的音乐检索框架

2.1 哼唱音乐的特征分析

哼唱音乐检索技术的关键就是从原始音乐数据中抽取出乐曲特征,根据这些特征实现快速准确的检索效果.音乐最为显著的特征为音乐的语句特征.

定义 1 音乐的语句.数据库中某首歌曲 $X = (x_1, x_2, \cdots, x_{m-1}, x_m)$,哼唱查询序列 $Q = (q_1, q_2, \cdots, q_{n-1}, q_n)$.其中 x_1, \cdots, x_m 为歌曲 X 的音乐语句, q_1, \cdots, q_n 为哼唱片段 Q 的音乐语句,下标 m 和 n 表示音乐 X 和哼唱片段 Q 中最后一句音乐语句序号.

音乐具有语句特征,而人们哼唱歌曲时,通常是以音乐语句为单位进行哼唱和自然停顿.如图 1 所示,可以看到这首歌曲由多句音乐语句构成,人们在哼唱时,句尾会自然停顿.

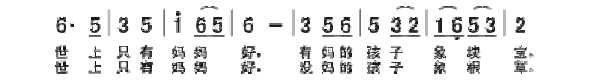


图 1 儿歌“世上只有妈妈好”简谱
Fig. 1 Song “mom is the best in the world” numbered musical notation

但是,传统的音乐检索技术只是将音乐的音高序列看作普通的时间序列来处理.通常采用传统的全序列匹配方式^[2-3]和子序列匹配方式^[4-5]进行查询匹配.全序列匹配是将数据库音乐和哼唱音乐划分成等长的一个个音乐片段,如图 2 所示,用每一条哼唱片段 q_1, q_2, q_3 分别与音乐库中的片段 $x_1, x_2, x_3, x_4, x_5, x_6$ 进行比对,找出相似度最高的音乐片段,从而得出查询结果.而在子序列匹配方式中,是将哼唱片段在音乐库中的音乐序列上进行滑动比对,如图 3 所示,哼唱音乐 Q 在数据库中的某首歌曲 X 从左向右滑动匹配,在歌曲 X 中找出相似度最高的音乐片段,如果此音乐片段属于相似度最高的前 K 个歌曲,则 X 即为查

询结果.

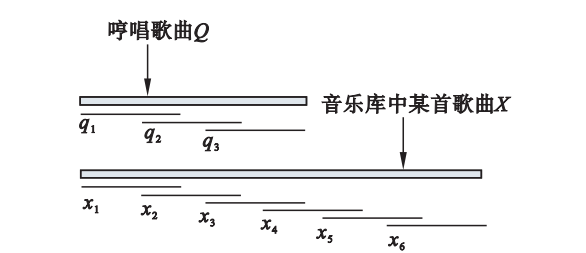


图 2 全序列匹配音乐片段划分示意图
Fig. 2 Example of partitions of music sequence based on whole matching

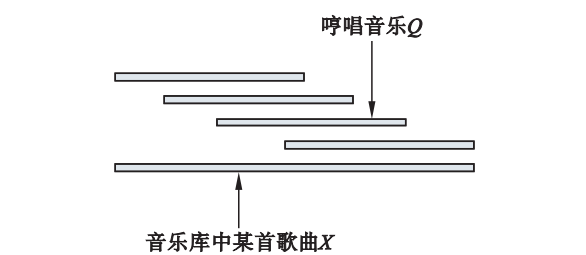


图 3 子序列匹配哼唱音乐检索示意图
Fig. 3 Diagram of query by humming based on subsequence matching

将数据库中的音乐信息和哼唱旋律按音乐语句划分,具体划分如图 4 所示.图 4 中,哼唱歌曲 Q 按哼唱自然停顿将划分为哼唱音乐语句 q_1, q_2 ,音乐库中的歌曲也按音乐语句的形式划分为 x_1, x_2, x_3, x_4 .使 q_1, q_2 分别与音乐库的音乐语句片段 x_1, x_2, x_3, x_4 进行比对,这样符合人们哼唱习惯的匹配方式,使检索过程具有更高识别率和效率.

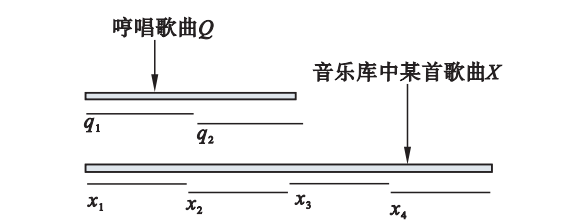


图 4 基于语句特征的音乐片段划分示意图
Fig. 4 Example of partitions of music sequence based on sentence features

基于上述音乐特征,本文的音乐哼唱检索系统整体框架由 3 个模块构成,分别是旋律提取模块、特征数据库模块、旋律匹配模块,如图 5 所示.

2.2 旋律提取模块

旋律提取模块的主要作用是从用户的哼唱信息中提取旋律特征,并且转化为可以用来搜索的特征向量.提取基于原始哼唱的基频曲线,保留了音乐旋律的原始曲线,避免了音符识别和切分可能出现的错误.本文在旋律提取方面将 wav 文档转化为音高序列.使用者哼唱后所录制的 wav 档案,必须经由音高追踪器算出声音讯号的频率,接

着再使用式(1)将频率转为半音差,其中,semitone 为半音差,freq 为声音讯号频率.

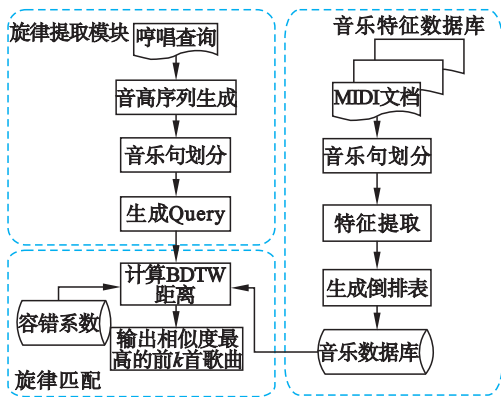


图 5 基于语句特征的音乐检索框架

Fig. 5 Workflow of query by humming based on sentence features

$$\text{semitone} = 12 \times \text{lb}\left(\frac{\text{freq}}{440}\right) + 69. \quad (1)$$

本文的工作重点为哼唱片段与音乐数据库中旋律的快速检索技术研究,所以在旋律提取方面不作过多讨论.

2.3 音乐特征数据库模块

特征数据库模块主要任务是建立带有音乐语句信息的特征数据库.在特征数据库中,为每首歌曲建立一个歌曲文件,歌曲文件的内容同样按音乐语句划分.进行处理之后,所有的旋律片段将按照 3 元组 (ID, Name, Data[]) 进行存放:ID 为数据库中音乐的序号;Name 指示歌曲的名字;数组 Data[] 为具有语句信息的音高序列.将休止符判定为语句的结尾,作为语句划分的依据.

2.4 旋律匹配模块

数据库中的某音乐语句 x , 对应的哼唱音乐语句 q . 假设音乐语句 $x = 7, 5, 3, 5, 2, 6, 5, 6$; $q = 7, 5, 5, 3, 5, 6, 6, 6$. q 与 x 相比,有增漏音符和唱错音符的现象,这在哼唱过程中,是很难避免的.因此在旋律匹配过程应具有一定的容错能力.但是也应避免音乐语句的一小段序列跟哼唱音乐语句中一大段进行匹配的病态匹配情况.因此本文提出一种新的相似度度量方式 BDTW 算法,具体方法将在第 3 节中做详细介绍.

3 基于语句特征的音乐检索技术

3.1 基于语句特征的音乐检索算法基本思想

在传统的哼唱识别匹配方法中,用 DTW^[2] 距离会出现一个音高序列中的一小段序列跟另一个音高序列中的一大段进行匹配的情况,这通常

不是想要的结果.基于此,本文提出一种新的序列匹配方法——限制性容错动态时间规整算法 BDTW. 用户更需要根据自己情况,设置容错限制系数 α ,用于限制在哼唱查询片段 Q 和数据库中音乐片段 X 中允许出现的容错程度.

BDTW 算法是以 DTW 算法思想为基础,并用 α 系数限制在哼唱查询片段 Q 和数据库中音乐片段 X 中允许出现的容错程度.即,用户可以根据自己哼唱水平的特点设置 α 系数,而 α 与 L 的乘积表示相对于音乐片段允许的最大间隔长度,如图 6 所示, L 为音乐语句的长度.即对 DTW 弯曲路径的计算附加了新的条件,使得哼唱音乐与目标音乐匹配时,音高序列下标之间的距离小于 $\alpha \times L$,为它确定了弯曲路径所能到达的范围,即哼唱匹配时容错范围.

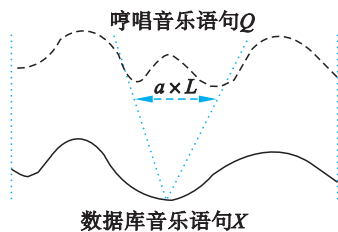


图 6 BDTW 距离弯曲路径示意图

Fig. 6 Warping path of BDTW

$$\left. \begin{aligned} \text{BD}(i, j) &= \min \left\{ \begin{aligned} &\text{BD}(i-1, j-2) \\ &\text{BD}(i-1, j-1) \\ &\text{BD}(i-2, j-1) \end{aligned} \right\} + \text{dist}_{\text{bound}(\alpha)}(i, j); \\ \text{dist}_{\text{bound}(\alpha)}(i, j) &= \begin{cases} \text{dist}(i, j), & \text{if } |i-j| \leq \alpha \times L, \\ \infty, & \text{if } |i-j| \geq \alpha \times L. \end{cases} \end{aligned} \right\} \quad (2)$$

定义 2 限制性容错动态时间规整距离. 给定音乐语句 Q 和 X , 语句长度为 L , 容错限制系数为 α , 它们的 BDTW 距离如式(2)所示. i 表示哼唱音乐音高序列的指针, j 表示目标歌曲音高序列的指针, 而 $\text{dist}(i, j)$ 则表示两者间的欧几里得距离, $\text{BD}(i, j)$ 为限制性容错动态时间规整距离.

BDTW 计算时的比对路径 W 是由元素 $w_1 = (Q_1, X_1)$ 到 $w_K = (Q_L, X_L)$ 的连续元素构成. W 的第 k 个元素记为 $w_k = (i, j)_k$, 表示第 k 个元素由点 Q_i 和 X_j 匹配构成. 这样 1 条完整的比对路径为 $W = w_1, w_2, \dots, w_k, \dots, w_K$, 类似于 DTW 比对路径, 此 BDTW 比对路径应满足下面三条性质:

性质 1 端点对齐: $w_1 = (1, 1)$ 和 $w_K = (L, L)$, 两个哼唱语句序列的开始位置和结束位置要分别对齐.

性质 2 路径连续: $w_k = (i, j)_k$, 那么 $w_{k-1} = (i', j')$, 其中 $i - i' \leq 1, j - j' \leq 1$, 即音乐语句比对

路径的计算每次只能走向矩阵中相邻的元素。

性质 3 单调性: $w_k = (i, j)_k$, 那么 $w_{k-1} = (i', j')$, 其中 $i - i' \geq 0, j - j' \geq 0$, 即音乐语句比对路径的计算总是沿着时间轴单调向前移动。

3.2 基于语句特征的音乐检索的索引结构

根据音乐数据特征, 本文提出建立支持语句特征的音乐检索索引结构 DIS 的方法, 使得检索阶段只需哼唱音乐语句与音乐数据库中的部分音乐语句进行相似度计算, 达到快速检索的目的。

由于音乐风格不同, 音乐语句旋律又具有不同特点, 首先在音乐语句集合 M 中分别计算两两音乐语句之间的 BDTW 距离, 然后从音乐语句集合中找出距离最大的两个音乐语句, 即最不相似的音乐语句 x_1 和 x_2 。然后对集合 M 中除了 x_1 和 x_2 之外的音乐语句分别与 x_1 和 x_2 作比较, 若音乐语句与 x_1 的距离小于 x_2 , 则将它归入集合 M_1 ; 反之归入集合 M_2 , 从而将集合 M 分成集合 M_1 和集合 M_2 两部分。同样的方法, 再分别对集合 M_1 和集合 M_2 划分为更小的集合。如此递归下去, 即可对数据库中的所有音乐语句建立树形索引, 如图 7 所示。

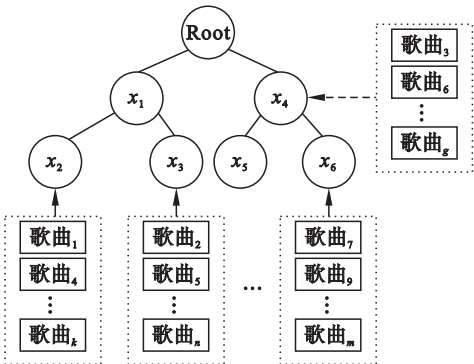


图 7 DIS 索引结构
Fig. 7 DIS index

检索过程为: 1) 假设用户哼唱输入的音高特征序列为 Q , 其长度为 L ; 2) 生成 Q 的所有音乐语句集合 G ; 3) 对于 G 中的任意音乐语句 q_i , 将 q_i 对应的满足条件差异小于 e 的所有链表中的信息加入集合 A , 其中 e 为长度差异阈值; 4) 对 G 中的另一个音乐语句 q_j , 如果在集合 A 中某音乐 E 也存在, 则将其相似度 $\text{sim}(Q, E)$ 加 1; 5) 对 $\text{sim}(Q, E)$ 按照从大到小的顺序排序, 将其中前 K 个音乐语句及其所在歌曲的序号加入结果集 C 中。

4 实验分析

为了测试本文所提检索技术的有效性, 本文

在 2 个真实的数据集 (MIREX 竞赛提供的语料库) 上进行了实验。MIR - QBSH 数据集共有 2 048 首 MIDI 格式的歌曲, 有查询片段 4 431 个, 歌唱者从歌曲歌首开始歌唱。IOACAS 数据集中, 有 298 首 MIDI 的格式音乐和 759 个查询片段, 歌唱者从歌曲任意位置开始歌唱。测试资料的取样频率 8 kHz, 单声道 (mono), 8 位分辨率的 wav 格式储存, 经由音高追踪器转换成音高序列。所有实验均在主频为 2.66 GHz 的 Intel Core Q8400 CPU 和内存为 2 GB 的 PC 机上完成, 使用的操作系统是 Ubuntu 12.10, 64 位, 编程语言采用 C++。

4.1 哼唱音乐语句划分检索效果检测与分析

首先对音高序列提取出语句特征, 将音乐按语句进行划分, 以音高序列中连续出现音高为 0 的位置视为一个语句断句。用户哼唱音高一般较高或者低于原始音乐, 需要对用户的哼唱音高和数据库中的歌曲同时做归一化处理, 则将每一个数据点都更新为其原来值与平均高度的比值。

本文首先在 MIR - QBSH 和 IOACAS 两个数据集上对 BDTW 算法进行划分音乐语句检索有效性和检索速度的效果测试对比, 其识别率和效率如图 8、图 9 所示。容错限制系数 α 取值应由用户跟自己哼唱情况给出, 本文在测试时, α 取值为 10%。本文按数据集中人们哼唱的语句个数, 从只哼唱 1 句到哼唱 5 句, 一共划分为 5 种情况, 由图 8、图 9 可以看出在有效性方面, 以音乐语句为单位进行检索在 5 种情况下都具有更好的性能, 特别是随着语句个数的增多, 这种优势更加明显。这主要是因为按句划分更符合人们哼唱的规律, 划分为音乐语句进行匹配避免了因为没划分音乐语句时前面的错位匹配影响了后面匹配的效果, 使得效率明显提高。

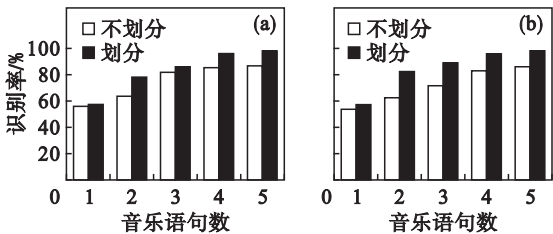


图 8 音乐划分语句检索识别率测试效果
Fig. 8 Retrieval precision test results of partition
(a)—数据集为 MIR - QBSH; (b)—数据集为 IOACAS.

4.2 基于语句特征的音乐检索算法检测与分析

本节实验在数据集上对 BDTW 算法与同类算法 DTW 进行比较。容错限制系数 α 取值为 10%，本文对比了在 DIS 索引结构下, 返回结果

为 top - 1, top - 5, top - 10, top - 15, top - 20 时, BDTW 算法与 DTW 算法在运行有效性和运行速度的效果测试对比. 如图 10, 图 11 所示, BDTW 算法具有更有效的哼唱音乐检索结果. 在 DIS 索引结构中, 查询识别率和效率随着 k 值的增加而增大, 并且 BDTW 的性能明显优于 DTW. BDTW 在 top - 5 查询时, 已经取得了很高的识别率. BDTW 算法具有更有效的哼唱音乐检索效率, 但是总的查询时间基本稳定.

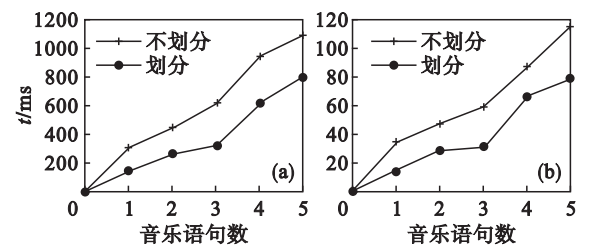


图 9 音乐划分语句检索效率测试效果
Fig. 9 Retrieval efficiency test results of partition
(a)—数据集为 MIR - QBSH; (b)—数据集为 IOACAS.

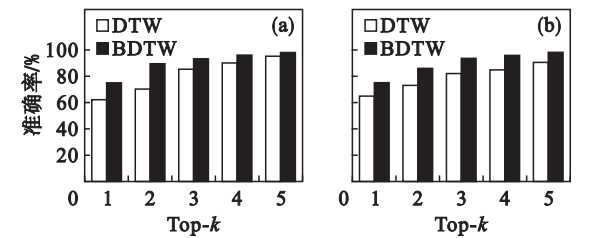


图 10 BDTW 算法检索准确度测试效果
Fig. 10 Retrieval precision test results of BDTW
(a)—数据集为 MIR - QBSH; (b)—数据集为 IOACAS.

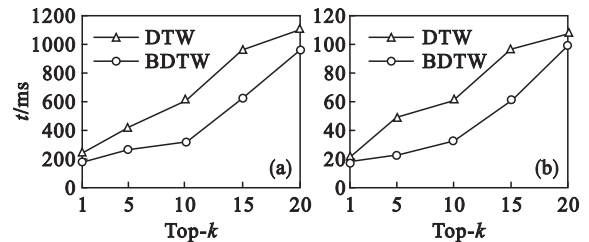


图 11 BDTW 检索效率测试效果
Fig. 11 Retrieval efficiency test results of BDTW
(a)—数据集为 MIR - QBSH; (b)—数据集为 IOACAS.

5 结 论

本文提出了一种新的基于语句特征的音乐哼唱快速检索技术. 其中 BDTW 算法允许用户根据自己哼唱水平给出容错限制系数, 限制数据库和查询序列的局部最大差异长度, 对音乐语句进行全序列匹配. 另外, 本文提出索引结构 DIS, 通过建立索引结构, 从而减少查询时间, 达到快速检索的目的. 实验结果表明本文提出的方法能够快速有效地为音乐哼唱问题返回正确的查询结果.

参考文献:

[1] Ghias A, Logan J, Chamberlin D, et al. Query by humming: musical information retrieval in an audio database [C]// Proceedings of ACM International Conference on Multimedia. San Francisco: ACM, 1995: 231 - 236.

[2] Zhu Y, Shasha D. Warping indexes with envelope transforms for query 2 by 2 humming [C]// Proceedings of Special Interest Group on Management of Data. San Diego, 2003: 181 - 192.

[3] Unal E, Chew E, Georgiou P, et al. Challenging uncertainty in query by humming systems; a fingerprinting approach [J]. Transactions on Audio Speech and Language Processing, 2008, 16(2): 359 - 371.

[4] Kotsifakos A, Papapetrou P, Hollmén J, et al. A subsequence matching with gaps range tolerances framework; a query by humming application [C]// Proceedings of Very Large Data Base. Seattle, 2011: 761 - 771.

[5] Kotsifakos A, Papapetrou P, Hollmén J, et al. Hum-a-song: a subsequence matching with gaps-range-tolerances query-by-humming system [C]// Proceedings of Very Large Data Base. Istanbul, 2012: 564 - 574.

[6] Yang X C, Qiu T, Wang B, et al. Negative factor: improving regular-expression matching in strings [J]. Acm Transactions on Database Systems, 2016, 40(4): 1 - 46.

[7] Yang X C, Wang Y, Wang B, et al. Local filtering: improving the performance of approximate queries on string collections [C]// ACM SIGMOD International Conference on Management of Data. Melbourne: ACM, 2015: 377 - 392.

[8] Wang P P, Wang B, Luo S Y. Top-K similarity search for query-by-humming [C]// Web-Age Information Management. Nanchang, 2016: 198 - 210.