哼唱检索中基于分段信息的匹配算法研究

Segmentation based Melody Match in Query-by-Humming

（申请清华大学工学硕士学位论文）

培 养 单 位 ： 计算机科学与技术系

学　　科 ： 计算机科学与技术

研 究 生 ： 曹 文 晓

指 导 教 师 ： 郑 方 研 究 员

二〇一〇年六月

摘 要

哼唱检索是基于音乐内容的检索，比传统的基于描述信息的音乐检索更人性化。

在哼唱检索系统中，用户通过麦克风等音频输入设备哼唱几句歌词，接着系统通过提取一定的特征，将哼唱语音特征与预先建立的旋律特征数据库中的特征进行比对并排名，给出检索结果。

目前哼唱检索的旋律匹配算法主要有字符串匹配、编辑距离、HMM、线性伸缩、动态规划、指纹匹配等，其中基于线性伸缩的方法效果较好。对于基于线性伸缩的方法，线性伸缩参数包括拉伸系数和音高偏移的估计依然是难点，另外当旋律较长时由于不同局部的线性伸缩参数不同，线性伸缩方法应用统一的线性伸缩参数效果较差。

本文提出基于极值点分段信息和基于停顿点分段信息来解决上述问题。极值点分段信息是指由旋律中最高点对旋律进行划分构成的分段结构，通过极值点分段信息增加启发式估计线性伸缩参数时的候选起点来提高对拉伸系数和音高偏移估计的准确性。停顿点分段信息是指通过哼唱过程中的停顿位置对旋律构建的分段信息，通过停顿点分段信息对旋律采用动态规划或递归匹配的方法进行分段地线性伸缩匹配，可以达到更好的匹配效果。本文通过多项实验验证了所提出的两种分段信息的有效性。

本文使用 355 大小的哼唱数据库和 5223 大小的 MIDI 旋律数据库进行实验来评估新算法的改进。实验结果表明，在最好的情况下，基于停顿点分段信息的分段匹配算法比传统的 LS、DP、RA 算法 Top-1 分别提高 17%、14.7%和 4.8%，相应的 MRR 分别提高 15.3%、12.9%和 4.8%。另外，使用 RALS 作为精确匹配算法时新的启发式估计算法使系统的 Top-1 和 MRR 准确率分别提高了 1.8%和 3.3%。

实验结果验证了极值点分段信息和停顿点分段信息的有效性，同时也说明了基于停顿点分段信息的分段匹配算法是比传统线性伸缩算法更有效的算法。

关键词：哼唱检索　　线性伸缩　　动态规划　　递归匹配

Abstract

Query-by-Humming(QBH) is a content based retrieval method, which is more personalized designed than the conventional one based on metadata.

In QBH system, a user hums several phrases recorded by audio recording device. The system then extracts melody features from the humming speech, compares them with the features from a pre-built melody database, and finally gives Top-N results.

Currently, there are many algorithms for melody matching, such as Fast String Match, Edit Distance, HMM, Linear Scaling, Dynamic Time Warping, Fingerprint and so on. Most of the research work conducts based on the liner scaling method. But it still remains two difficult problems to many researchers, one is the estimation of parameters of tempo ratio and key transposition, the other is the matching result gets worse under the identical parameters when the humming query lasts longer.

In order to solve these two problems, we proposed a two segmentation information method which used the information from the extreme point and singing pausing point. The information is defined as segmentation information, which describes the phrases segmented by extreme points or singing pausing points.

Extreme point adds new intial parameters to heuristic parameter estimation so that the system can achieve more accurate estimation. Singing pausing point is useful for phrase-level melody matching with Dynamic Programming or recursive alignment for more efficient scoring. In this thesis, we did many experiments to prove the efficiency of the proposed segmentation information.

We used a query database with 355 humming queries and a MIDI database with 5223 MIDIs to evaluate the improvement of the new heuristic parameter estimation and phrase-level melody matching algorithms using the proposed segmentation information. It is shown that the new phrase-level melody matching algorithm has 17%, 14.7% and 4.8% improvement in Top-1 accuracy relative to LS, DP and RA methods at most. The corresponding improvements in MRR are 15.3%, 12.9% and

4.8%, respectively. At the same time, the new heuristic parameter estimation improved the Top-1 and MRR performance by 1.8% and 3.3%.

Thus the segmentaion information based on extreme point and singing pausing point is efficient for meldoy matching in QBH, and phrase-level melody matching algorithms are more efficient algorithms.

Keywords: Query-by-Humming Linear Scaling Dynamic Programming Recursive Alignment

目 录

第 1 章 引言 1
1.1 研究背景与研究意义 1
1.2 旋律匹配算法的研究现状 3
1.3 本文的研究内容及主要贡献 8
第 2 章 哼唱检索的基本原理 10
2.1 本章引论 10
2.2 MIDI 文件格式与旋律表示 10
2.3 音符与基频之间的关系 13
2.4 哼唱语音旋律特征的提取 16
2.4.1 音高的提取 16
2.4.2 音符的提取 19
2.5 哼唱旋律与 MIDI 旋律的匹配 23
2.5.1 匹配的定义 23
2.5.2 匹配的特征 23
2.5.3 匹配的难点 23
2.5.4 匹配的分类 25
2.6 哼唱旋律与 MIDI 旋律的匹配算法 25
2.6.1 基于字符串匹配的方法 25
2.6.2 线性伸缩 26
2.6.3 动态规划 27
2.6.4 基于递归的线性伸缩方法 28
2.7 本章小结 30
第 3 章 分段信息 31
3.1 本章引论 31
3.2 基于极值点的分段信息 32
3.2.1 极值点的定义 32
3.2.2 极值点的估计 34

3.3 基于停顿点的分段信息.....35
3.3.1 停顿点的定义.....35
3.3.2 停顿点的物理意义.....37
3.3.3 停顿点的估计.....39
3.3.4 估计参数的确定.....41
3.4 实验.....42
3.5 本章小结.....44
第 4 章 基于分段信息的匹配算法.....45
4.1 本章引论.....45
4.2 基于极值点分段信息的线性伸缩参数估计.....45
4.3 基于停顿点分段信息的动态规划.....49
4.4 基于停顿点分段信息的递归匹配.....51
4.5 系统结构与实验数据.....53
4.6 实验验证及结果分析.....56
4.6.1 评价准则及各算法描述.....56
4.6.2 基于停顿点分段信息的精确匹配.....57
4.6.3 候选分段点数对动态规划分段匹配的影响.....59
4.6.4 递归分段匹配准确率与递归层次的关系.....60
4.6.5 基于极值点的启发式线性伸缩参数估计算法.....61
4.6.6 实验小结.....62
4.7 本章小结.....62
第 5 章 结论与展望.....63
参考文献.....66
致谢与声明.....70
个人简历、在学期间发表的学术论文与研究成果.....71

第1章　引言

1.1　研究背景与研究意义

音乐检索作为与越来越庞大的音乐数据库的交互接口，在多媒体信息检索中起着至关重要的作用。传统音乐检索利用歌曲描叙信息(metadata)如歌名等，对描述信息进行索引并建立数据库，当用户通过文本查询时，按照文本检索的方式检索数据库中与查询文本相关度最大的数据返回给用户。

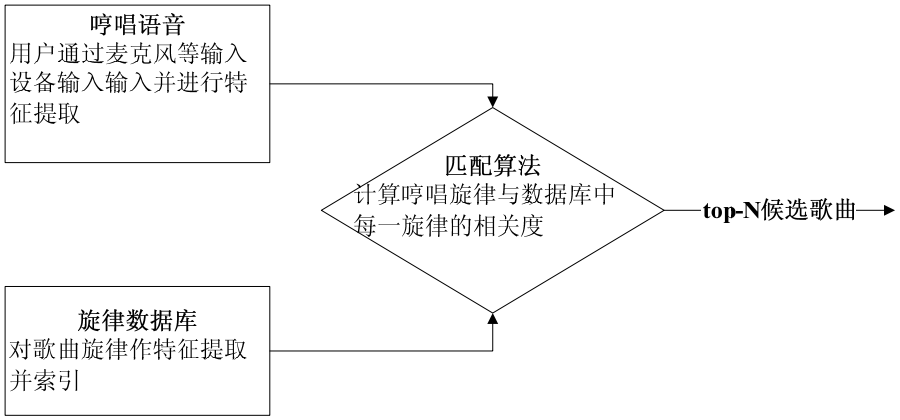

图 1.1　哼唱检索系统的一般结构

哼唱检索是基于内容的音乐检索方式，如图 1.1 为一般的哼唱检索系统的系统结构，检索系统主要包括三部分：

（1）旋律特征数据库

旋律特征数据库是哼唱检索系统的模板库，系统对原始的音乐数据进行特征提取，然后将特征索引并储存到特征数据库中。一般的哼唱检索系统使用 MIDI 数据库作为旋律特征数据库。

（2）用户输入与特征提取

这是系统与用户交互的接口，当用户检索音乐时，通过麦克风等录音设备哼唱几句歌词，一般为几秒到几十秒不等。哼唱形式可以是带歌词哼唱或以哒哒哒、吹口哨等方式哼唱。录音完毕后，系统对哼唱语音进行旋律特征提

取，并输入到匹配模块中。

（3）旋律匹配与候选输出

提取用户哼唱语音的旋律特征后，系统将该特征与数据库中的旋律特征进行比对，最后给出与用户查询相关度最大的前 N 条记录。

哼唱检索系统相比传统的音乐检索系统的优势有：一，更加精准；传统的音乐检索引擎通过歌曲的歌名，歌手，歌词等信息作为关键词来检索，随着流行音乐歌曲数量的快速增长，经常会有同歌名的歌曲出现，同一个歌手的歌曲数目也较多，尽管可以按专辑划分，要记住每一专辑中的歌名并非易事。要精确地记住歌词也比较困难，同时不同歌曲的歌词也可能出现重复，通过歌词检索时若输入歌词太少则很难精确地找到目标歌曲。因此通过传统的音乐检索引擎检索歌曲具有一定复杂度，同时必须结合一定的检索技巧。而哼唱检索则不同，一首歌的旋律一般比较独特，尤其歌曲中高潮或旋律优美的部分更易于记忆。由于旋律的“唯一性”，当用户将对应旋律以哼唱或唱的方式输入到哼唱检索系统时，系统便可以通过一定的匹配算法，找出对应的歌曲。因此，在哼唱检索系统的算法足够好的条件下，哼唱检索方式比传统检索方式更加精准。二，更加人性化；通过哼唱检索的方式，用户仅需要麦克风或者其他录音设备，通过语音输入方式，来进行检索。而传统检索方式需要通过键盘等文字设备输入文字进行检索。就普通用户而言，文字输入的方式比语音输入的方式复杂，并且输入速度慢，因为文字输入受用户使用的输入法、盲打等计算机水平的影响，而语音输入与平常说话类似。因此哼唱检索的方式比传统的音乐检索方式更加人性化，更易于使用。

从提出哼唱检索概念开始，哼唱检索就一直是国内外研究的重点课题。最早的哼唱检索研究由 Ghias 在 1995 年发表的一篇论文[1]开始，文中使用了较简单的方法，将旋律的升高降低和保持不变转化成字符串并进行字符串匹配。由于这种旋律表示方法过于粗糙，需要的哼唱语音比较长，在 Ghias 的论文中使用的数据库为 183 首。McNab 则实现了第一个可以通过互联网进行哼唱检索的系统[2]，尽管使用了字符串匹配方法，在特征和数据库上却有进一步的研究。台湾清华大学的张智星也在哼唱检索方面做了很多研究工作[3-5]，通过研究线性伸缩及动态规划等方法，开发出了多个哼唱检索系统。上海交通大学的李扬等通过使用近似旋律匹配即线性对齐方法进行哼唱检索，并构造了哼

唱检索系统的原型，其中数据库中使用了 3864 首歌曲[6]。Shifrin[7, 8]以及国内深圳大学的陈知困[9]等研究了使用 HMM 方法进行哼唱检索匹配。还有加利福尼亚大学的 Unal 尝试使用类似指纹的方法进行哼唱检索[10]。总体而言，目前的研究方法主要包括字符串匹配、编辑距离、动态规划、HMM、线性伸缩、指纹等方法。由于算法以及数据等方面的原因，哼唱检索目前还难以达到实际应用的要求。

尽管哼唱检索系统比传统音乐检索方式在本质上有很大优势，要取代传统音乐检索方式尚为时过早。在哼唱检索的实验环境下，使用较小的数据库（这里指几百到几万不等）可以得到较好的准确率，但实际应用时会遇到一定的困难。一者，数据库的收集难度较大。目前多数算法的研究是基于 MIDI 音乐格式，而对于 MIDI 音乐格式如果要建立全面的数据库需要通过手工录入的方式，增加了数据库的工作量，同时数据库的更新维护也过于繁琐。二者，匹配算法的准确率难以达到实际应用要求。实际应用环境条件下，受各种因素如噪声、哼唱质量差等因素影响，匹配算法的准确率会大大下降。

目前哼唱检索还有较大的研究空间，如何进行更有效的旋律匹配依然是研究中的难点。本文的研究工作旨在以目前已有的匹配算法为基础，通过研究旋律中的分段信息并用于匹配，达到提高匹配效果的目的，为今后的哼唱检索研究提供有效的参考。

1.2　旋律匹配算法的研究现状

哼唱检索中的关键算法是计算哼唱旋律与数据库中旋律间的相似度，大多数对哼唱检索匹配的研究是基于哼唱旋律与 MIDI 旋律进行的。由于哼唱旋律和 MIDI 旋律间存在固有的差异，如 MIDI 中的三个音符可能在歌曲中对应 2 个歌词，因此哼唱语音经过提取后也是 2 个音符，这种特殊的性质要求哼唱检索的匹配算法对错误的容忍度较高。

国外主要的旋律匹配方法有字符串匹配、编辑距离、动态规划、线性伸缩、HMM、指纹等，大部分的研究工作基于某一种方法或多种方法结合的思路进行。

字符串匹配的方法是最早用于哼唱检索的方法，其主要思想是将旋律表示为字符串，然后通过字符串检索、快速匹配等方法进行匹配。快速字符串匹

配已经有许多成熟的方法，但应用于哼唱检索时须考虑一定的错误容忍，即在匹配结果中能容忍一定数量的错误匹配。目前已有不少对可容忍 k 次错误的字符串匹配方法的研究[11-14]。对于可容忍 k 错误的快速字符串匹配算法，基于 Boyer-Moore 的字符串匹配算法被认为是实际应用中最好的算法，同时该方法的代码相比暴力搜索而言更简单[14]。文献[1]使用 Baesa-Yates 和 Perleberg 提出的可容忍 k 次错误的字符串匹配算法[12]进行哼唱检索，在含 183 首歌曲的 MIDI 数据库上用 10~12 个音符构成的字符串序列进行检索，能够达到 90%的准确率。使用字符串检索的方法对特征提取的要求比较高，文献中该方法特征提取占用的时间较大。McNab[15]在哼唱检索中使用音高、节奏、旋律的上升下降曲线等特征来进行检索，并且只从歌曲的起始点使用字符串匹配进行检索，同时研究了从数据库中检索到正确歌曲所需要的特征数量如音符个数，是否使用节奏信息等，还研究了所需要的音符个数随数据库大小变化的关系。通过字符串快速匹配的方法进行哼唱检索，优点是较直观，匹配速度较快，缺点是特征提取的难度及要求比较高。由于字符难以表示旋律变化的丰富性，假如以字符表示绝对音高，则哼唱旋律与数据库旋律间存在整体的音高偏移，因而表示的字符串不一致的概率大大增加。而对于以字符表示音高变化，若单纯表示音高的升降或保持不变，则过于简略，很难将目标旋律与其他旋律区分开，导致返回数据集过大。而若以字符表示音高变化的不同程度，则存在程度分界的问题，两个不同的音高变化值极相近，但可能被归类成不同音符，导致错误。

编辑距离又称 Levenshtein 距离，用于计算一个字符串转化为另一个字符串所需要的最少编辑操作次数，一般使用动态规划方法计算。Prechel[16]使用类似快速字符串匹配的方法，将旋律根据音高的升降和保持不变转化为 U/D/S 表示的字符串，然后通过从数据库中检索与哼唱旋律的特征字符串间编辑距离最小的歌曲作为匹配结果。通过在 Parsons[17]数据库的 10370 首古典音乐数据库上使用 106 个录音片段进行哼唱检索，得到 Top-1 正确率为 44%，Top-40 正确率为 77%，并指出大部分的错误是由于呼吸导致，如果能在录制哼唱语音时不呼吸，则上述对应的准确率可提高到 59%、86%。编辑距离的方法与字符串匹配类似，相比可容忍 k 错误的字符串匹配，编辑距离可以容忍插入、删除及替换三种错误类型，比可容忍 k 错误的字符串匹配更鲁棒。缺点也与字符串匹配类似，对旋律的表示过于简化，同时基于动态规划来计算编辑距

离的耗时也比快速字符串匹配大。

动态规划是计算机科学中常用的用于求解可分解为子问题的最优化方法，它将求解当前问题最优解的问题，转化为求解子问题的最优解问题。因此不少哼唱检索匹配算法的研究都基于动态规划的方法[3, 18-20]。Jyh-Shing Roger Jang 使用多次的动态规划并同时估计音高偏移以达到最好的匹配效果，在估计音高偏移时使用启发式估计算法。通过在 800 首歌曲的数据库上使用 200 个哼唱片段进行测试，每两段旋律间的匹配调用 5 次动态规划，在限制哼唱旋律均从歌曲的起始部分开始哼唱并哼唱 5 至 8 秒的条件下，得到 Top-1 的准确率为 76%，Top-20 为 86.5%，说明这种基于动态规划的方法能够满足一般哼唱水平的人的使用要求。动态规划方法的优点是对匹配旋律的要求不严格，抗噪性强，可以通过搜索不同路径达到最优匹配结果，缺点是匹配时间长，计算量大，数据库增大时系统时间响应的变化尤为明显。

考虑到哼唱旋律与标准 MIDI 旋律间存在的线性关系，线性伸缩的方法被引入到哼唱检索的旋律匹配算法中，线性伸缩方法通过将其中一旋律的曲线进行横向的拉伸和纵向平移来与另一旋律曲线对齐，最终调整到最好的对齐效果计算分数。Jyh-Shing Roger Jang 提出使用线性伸缩匹配的方法作为距离函数并利用树结构搜索哼唱旋律的最近邻作为检索结果[4]，通过使用普通哼唱水平的人的 1000 个哼唱语音在包含 3000 首歌曲的数据库上进行测试，Top-20 的准确率达到 73%，对应的系统响应时间为 2 秒。同时认为，旋律可能存在非线性的速度变化，这时使用动态规划能够达到更好的匹配效果，但动态规划的时间消耗较大，因此必须在系统准确率和时间响应上作适当的平衡。线性伸缩方法的优点是计算简单且直观，缺点是匹配起点的确定困难，一般假设哼唱旋律从句子的起始点或歌曲的起始点开始来降低搜索的空间。另一问题是线性伸缩的参数包括拉伸系数（Tempo variation）和音高偏移（Key transposition）的估计问题，由于没有直观的办法获取这两个参数，一般通过多次使用不同参数计算并选最优的方法来估计。再者，当哼唱旋律较长时，整体进行线性伸缩匹配的效果下降，因为哼唱旋律的局部可能存在不同的线性伸缩参数。

HMM 作为语音识别中的重要工具，同样也被用于哼唱检索的研究。HMM 全称是隐马尔科夫模型，用于描述隐含了未知参数的马尔科夫过程。使用 HMM 进行哼唱检索时，数据库中的旋律表示为 HMM 的模型，而查询旋律则

作为观察序列，当进行哼唱检索时根据每一数据库旋律输出该哼唱旋律的后验概率进行排序。Shifrin[8]将哼唱旋律表示成音符序列，定义 HMM 中的每一个状态具有 2 个属性，一是音高变化，即音符相对上一音符的音高差，二是时间变化，即音符的持续时间。状态间的转移概率则使用 MIDI 数据库训练得到。在哼唱检索时利用 HMM 的前向算法计算匹配的似然度作为匹配概率。通过使用 24 个哼唱语音在包含 277 首 MIDI 的数据库上进行检索，并和字符串匹配方法对比。结果表明，字符串匹配方法得到的 Top-1 和 Top-5 准确率分别为 16.7%和 20.8%，而 HMM 方法的对应准确率分别为 41.7%和 58.3%。该方法的局限是对于查询旋律长度大于 HMM 中的最长路径时会导致错误，同时查询旋律中存在个别音符丢失的情况时也易导致错误。通过使用更大的数据库进行进一步研究[7]，系统对于哼唱旋律越长的情况则检索结果越准确，同时对于不理想的哼唱旋律也能做较好的匹配。使用 HMM 的方法进行匹配的优点是引入了概率的概念，可以通过对现有数据的统计达到较好的匹配效果，缺点是需要训练模型，并且识别的准确率与训练数据的关系较大。

另外还有指纹识别（FingerPrint）的方法，指纹识别最早用于人的身份辨识，由于在哼唱检索中这一方法与身份辨识中的指纹识别方法相似，因此也称为指纹识别。文献[10]中使用 HMM 的方法提取旋律特征，将哼唱旋律切割成音符，并提取 RPD(relative pitch difference)和 RDD(relative duration difference)两种特征，然后取 RPD 和 RDD 曲线中的极值点来建立指纹样本。哼唱检索时通过计算哼唱旋律与 MIDI 旋律间在 RPD,RDD 上的平方差作为匹配分数，最终保留前 5 个候选。通过使用来自 80 人的 400 个哼唱查询语音及大小为 1500 的 MIDI 数据库进行哼唱检索，结果表明在受过音乐教育的人对应的哼唱数据集上可达到 88%的准确率，而对于未受过音乐教育的对应准确率为 70%。对应的使用传统的编辑距离匹配的方法得到的准确率分别为 86%和 62%。说明这种指纹识别的方法对于未受过音乐教育的人的哼唱查询具有更好的鲁棒性。指纹识别的优点是仅使用旋律中的少量但独特的信息，便于快速匹配和索引，缺点是确定有效的指纹信息并在哼唱旋律与 MIDI 旋律之间保持一致比较困难。相比身份识别中的二维指纹，哼唱检索中的指纹是一维信息，因而表示的信息量也较少，要做到精确匹配，必须联合同一旋律中的多个指纹进行计算。

国内对于旋律匹配算法的研究起步相对晚一些，主要的算法分类与国外的

算法大致相同，也包括字符串匹配、编辑距离、动态规划、线性伸缩及 HMM 的方法，目前对类似指纹识别的哼唱检索研究较少见。

字符串匹配仍是常用的方法[21, 22]，文献[21]首先提取歌谱轮廓特征，通过构造标准音调差值图将哼唱旋律表示为歌谱特征，然后通过动态规划方法计算歌谱字符串间的相似度。通过在 405 首歌曲的数据库上进行检索，得到的 Top-5 准确率超过 90%。

编辑距离的方法也被用于旋律匹配，文献[23]使用动态规划计算哼唱旋律和 MIDI 旋律间的编辑距离，通过在动态规划中引入模糊隶属度函数计算两音高差间的相似度，同时引入音长比信息进行相似度计算，最终以两种相似度的加权作为最终的编辑距离分数。通过在包含 2500 个 MIDI 文件的数据库上使用 90 个哼唱片段进行测试，结果表明算法相比音高差 5 阶量化方法在 Top-10 的准确率上由 59%提高到了 75%。

动态规划的方法继续用于哼唱检索匹配。和传统的哼唱语音检索 MIDI 旋律的方式不同，文献[24]考虑了哼唱语音和哼唱语音间的直接匹配，通过对哼唱语音提取 MFCC1~3 及过零率特征并进行适当简化，再与数据库中的哼唱语音特征使用 DTW 计算相似度。数据库中的候选歌曲都标注了句子起始点，因此取哼唱旋律与数据库中对应歌曲的多个候选片段中匹配分数最好时的分数作为与该歌曲的匹配分数。通过对 6 位哼唱者的 70 余次哼唱检索进行测试，排名在前 15 位的概率超过 60%，说明系统是有效的。除此之外，还有包先春提出两层的 DTW 算法[25]，罗凯等提出在 DTW 中同时考虑音高差和音长差作为代价函数[26]，马志欣等提出模糊集合的概念并在 DTW 中使用音高差和音长比加权作为代价函数的方法[23]等。

线性伸缩的方法也获得了较好的效果。中科院声学所的吴晓等人[27]使用线性伸缩的方法，通过对哼唱旋律和数据库旋律进行递归的自顶向下的分段匹配来进行检索。在确定要匹配的哼唱旋律片段和数据库旋律片段后，取数据库旋律的中间音符，根据预先设定多种线性伸缩系数和音高偏移系数的组合，按对应的划分比例确定哼唱旋律中的对应音符，将两段旋律划分成两组片段，并计算两组片段对应的线性伸缩匹配分数，取分数最优的一种对哼唱旋律的划分方式，如果已经达到设定的递归次数，则直接返回前述分数，否则继续对划分的两对子片段进行同样的匹配过程，直到达到预设的递归次数。通过简化该方法的计算过程，由该方法衍生出三种快速匹配的方法，用于快速过

滤。通过在包含 1180 首歌曲的 MIDI 数据库上，使用 875 个哼唱查询语音进行测试，实验结果表明该方法比 LS、DTW 方法具有更好的性能。同时指出当查询旋律的长度在 7~15 秒时具有较好的匹配准确率，大于 13 秒时结果变差，可能由于该递归方法在查询旋律过长时不再有效。另外也有李扬等人[6]同时考虑音高和节奏来进行线性伸缩匹配，也获得了较好的效果。

国内也有对 HMM 匹配方法的改进工作[9, 28]。中国人民大学的袁斌等人[28]通过对哼唱语音的音高差和音长比进行统计分析，并合成产生新的旋律和节奏训练数据库，同时对 HMM 的特征和训练方法都提出了改进。通过在包含 1500 个音乐片段的数据库上使用 27 个检索片段进行检索，前 5 位命中率达到 85.5%，得到了满意的效果。

还有中国人民大学的刘怡等人[29]研究了大型音乐哼唱系统中不同近似匹配算法的算法性能，并比较了后缀树、HMM、编辑距离、DTW 及单侧连续匹配（OSCM）的匹配方法。单侧连续匹配的方法采用 N-GRAM 的顺序哈希索引来加快查询处理速度。通过在包含 72000 首音乐片段的数据库上构造 1500 个不同类型错误的查询来比较其中 3 类方法，结果表明单侧连续匹配的方法查询速度快，在用户哼唱旋律只包含与旋律轮廓方向相同的错误时查询准确率能达到 100%，而哼唱包含两个以内与旋律轮廓方向相反的错误时，前 10 位的命中率在 90%左右，说明单侧连续匹配的方法是适用于大型哼唱检索系统的有效算法。

目前国内的研究起步晚，还有很大的研究空间。由于目前免费开放的中文相关的数据库资源较少，给不同研究工作间的比较带来一定困难。

1.3　本文的研究内容及主要贡献

旋律匹配算法一直是领域内的研究重点，在旋律匹配中利用的是音高特征（pitch），通过对音高特征序列中的音高使用特定算法进行分割可以得到音符序列，从而将旋律匹配的问题，归结为哼唱旋律的音高序列和音符序列与 MIDI 旋律的音符序列间的匹配分数问题。

线性伸缩方法作为一种效果比较好且计算简单的方法，面临的主要难点是线性伸缩参数的估计和局部参数精确化的问题。首先，进行线性伸缩需要确定两个必要参数，一是拉伸系数，即两旋律间的拉伸比例，二是音高偏移，

指对其中一旋律与另一旋律对齐最好时纵向平移的尺度。尽管估计这两个参数时可使用枚举的方法（如[27]中方法），但会导致大量计算，因此从速度角度考虑，枚举方法并不可取。另一种估计方法是通过启发式估计来确定最优参数，这种启发式方法在[5]中也已用过，只不过是基于动态规划的启发式方法。由于线性伸缩的匹配分数与它的两个参数间并非单调的关系，这种方法可能导致陷入局部最优点，不能获取全局最优参数，导致结果的准确率下降。另一个问题局部参数精确化的问题，当使用同一的线性伸缩参数进行精确匹配时，某些局部对齐得好，而某些则较差，不同局部需要的线性伸缩参数是不同的，如何对各部分使用相应的参数进行匹配，是精确匹配的关键问题。

为解决线性伸缩匹配中面临的上述问题，本文中提出了两种分段信息，一种是基于极值点的分段信息，极值点即旋律中的最高点，这种信息被用来增加启发式估计线性伸缩参数时的候选起点，提高估计参数的准确性。另一种是基于停顿点的分段信息，停顿点是哼唱过程中换气的位置，相当于音乐乐谱中的休止符。本文同时提出了使用停顿点的分段信息并利用动态规划以及递归匹配的方法来优化对 MIDI 旋律的分段匹配，从而实现哼唱旋律和 MIDI 旋律之间的分段匹配。

本文的主要贡献是：（1）提出两种分段信息特征；（2）利用极值点分段信息实现了一种基于极值点分段信息的启发式估计算法；（3）利用停顿点分段信息实现了两种分段匹配方法，一种使用动态规划的策略，另一种使用递归匹配的策略。本文最后给出的实验结果验证了所提出算法的有效性。

本文后续的内容安排如下：第二章介绍哼唱检索的基本原理，包括 MIDI 旋律、音符与基频的关系、特征提取及旋律匹配算法；第三章介绍极值点分段信息及停顿点分段信息；第四章研究利用分段信息进行匹配并提高检索的准确率，并研究动态规划和递归匹配两种优化策略；最后在第五章给出结论。

第2章　哼唱检索的基本原理

2.1　本章引论

本章阐述哼唱检索系统的基本原理。首先简单介绍 MIDI 文件格式及 MIDI 旋律的表示方法，接着通过阐述音符与语音基频间的关系，说明音符的音高与基频间存在固定的关系，而后以此为基础介绍了基频提取，基频到音符的转化方法，通过提取基频及音符来提取哼唱语音的旋律特征，最后介绍基于已提取的旋律特征的多种匹配算法，其中主要介绍了最早的字符串匹配方法及与本文研究相关的几种匹配方法，包括线性伸缩(LS)、动态规划(DP)、递归匹配(RA)三种方法。

2.2　MIDI 文件格式与旋律表示

本文的哼唱检索是基于哼唱旋律与 MIDI 旋律进行匹配的。MIDI 即乐器数字接口(Musical Instrument Digital Interface, 简称 MIDI)，是一项工业标准的通信协议，通常用于电子乐器等演奏设备，它定义了电子乐器演奏所需要的数据包括音调、音乐强度、节拍、节拍速度等。MIDI 标准是戴夫·史密斯于 1981 年向音频工程协会提出的，MIDI 规范的 1.0 版本发布于 1983 年。

MIDI 可以说是电子乐器的电子乐谱，MIDI 乐器演奏一个音符包括了三个要素：

（1）按下的 MIDI 乐器中的特定键

如中央 C，MIDI 乐器中的每个键对应了特定的 MIDI 音符编号，这指定了音符的频率，MIDI 音符编号可以参考 MIDI 标准[30]。

（2）按下音符键时的力度

用户按压键的力度越大，音符发声的强度就越大。

（3）释放按键的时间

释放按键和按下键间的时间差就是音符的时长。

若再通过一定的演奏速度弹奏各种音符，便构成了一首完整的音乐旋律。

MIDI 文件是 MIDI 旋律的数字表示，一个 MIDI 文件的内容由多个块(chunk)组成，每个块的数据格式如表 2.1 所示。

MIDI 文件中包含两种类型的块, 一种是头部块(Header Chunk), 用于描述整个 MIDI 文件的基本信息，对应块标识是 MThd，另一种是轨道块(Track Chunk), 用于描述 MIDI 的每个轨道的音乐数据，对应块标识是 MTrk。一个 MIDI 文件通常由一个头部块和多个轨道块组成，轨道块的数量取决于 MIDI 是单轨还是多轨及轨道的数量。如表 2.2 显示了 MIDI 文件由一个头部块和多个轨道块组成时的数据格式，其中各参数含义在**错误！未找到引用源。**中说明。

从表 2.2 中可看出，MIDI 文件主要用来表示 MIDI 中的事件信息，每一个轨道由多个事件构成，每一个事件设定了持续时间。有关 MIDI 事件的详细描述可参考[30]。表 2.4 中列出了几种常用的事件，由旋律轨数据格式及表 2.4 中的几种事件看出，MIDI 中每一轨的旋律演奏过程是发声、停止发声、…、发声、停止发声循环重复的过程，每次发声的过程都持续一定时间，且具有指定的音高。

表 2.1　MIDI 文件中的块格式

组成部分	描述
块标识	占 4 字节，ASCII 字符串，表示块的类型
头部数据长度	占 4 字节，表示当前块的长度，不包含前面 8 字节
数据	占用空间由头部数据长度描述，存放块的所有内容

表 2.2　MIDI 文件的块组成

	块类型	块长度	数据
MIDI 文件	MThd	6	<format><tracks><division>
	MTrk	<size>	<delta_time><event>,…,<delta_time><event>
			…
	MTrk	<size>	<delta_time><event>,…,<delta_time><event>

表 2.3　表 2.2 中的各变量说明

变量	说明
<format>	表示 MIDI 文件的格式，占 2 个字节，有效的数值是 0、1、和 2。0 表示 MIDI 文件只有一个轨道块，1 表示只有一个或多个轨道块，2 表示只有一个或多个独立的轨道块
<tracks>	表示轨道块的数量
<division>	表示<delta_time>的单位时间长度
<size>	表示块的长度，具体值由块内容的长度决定
<delta_time>	相对于前一个事件的时间长度
<event>	事件

表 2.4　MIDI 中的几种事件

事件	二进制格式	说明
开始发声	0x9n note speed	n 表示轨道号，可取 0~F, 对于多轨 MIDI 可同时演奏多个轨道，n 指定在哪一轨道发声，note 为音高的 MIDI 编号，如中央 C 为 60，speed 为按键速度，也表示音的强度。
停止发声	0x8n note speed	参数与开始发声一致，停止发声也可使用 0x9n note 0 来代替。
设置音量大小	0xbn 07 size 0xbn 39 size	7 表示设置主音量的高字节值，39 表示设置主音量的低字节值。

通过以上对 MIDI 文件格式的分析可看出，MIDI 音乐的每一轨旋律可以使用一条音符曲线表示。音符是乐谱中的基本单元，每个音符具有固定的音高，并且持续一定时间。如图 2.1 所示，表示了 MIDI 中某一轨的旋律，图中标注音符的音高为 52 即为对应的 MIDI 编号。在 MIDI 播放时，当播放到所标注音符时，音符在 7.8 秒发声，持续到 10.5 秒时停止发声，然后继续播放下一个音符，由此演奏完整的 MIDI 旋律。本文中我们主要研究单轨 MIDI 旋律，因此所有 MIDI 旋律均可以表示成如下的音符序列：

$$M = (d_1, p_1), (d_2, p_2), \cdots, (d_n, p_n) \qquad (2\text{-}1)$$

其中d_i表示第i个音符的持续时间，p_i表示第i个音符的音高值。

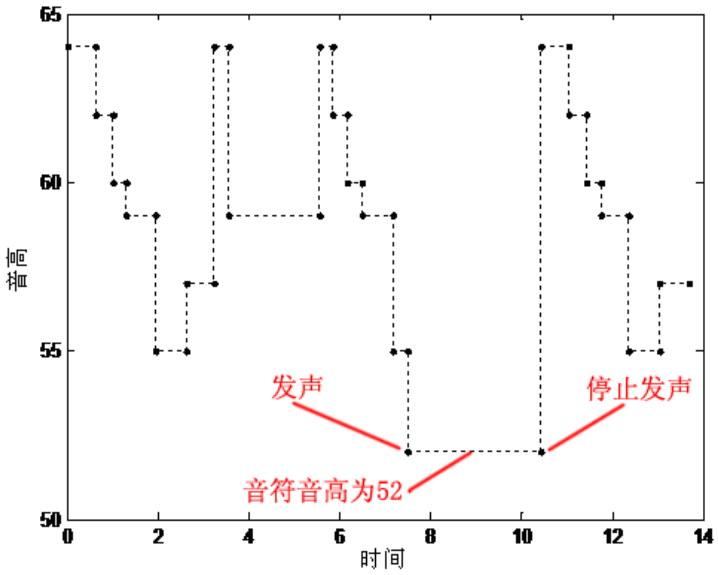

图 2.1　MIDI 旋律及音符的发声与停止发声事件

2.3　音符与基频之间的关系

在音乐中，通常会提到音高的概念，同时根据不同人发声的音高范围将男性分成男低音、男中音、男高音，将女性分为女低音、女中音、女高音。这种音高，即音调的高低实际上是语音信号基频大小的表现。尽管在语音信号中任何复合乐音实际上是由基频和许多谐波组成，通常我们所说的音调高低，是说它的基频的高低，而不管它的谐波成分[31]。

音高仅仅取决于基频大小，而音符作为旋律的基本组成元素，代表了一段时间内某一音高的声音。音符是西方音乐中的基本元素，是乐谱中构成音乐的最小单元，它使得音乐得以让人演奏、理解和分析。如果两个音符的频率相差整数倍，则听觉上会感觉非常相似，因此通常将这种频率相差整数倍的音符归入同一音高集合。两音符之间如果频率相差一倍的话，就称它们相差一个八度。因此在音乐中，要确定一个音符，必须知道它所在的音高集合及所在的八度。在传统音乐中，可以使用 7 个拉丁字符：A,B,C,D,E,F,G 来表示音高集合的分类，在不同的八度中，这些音符按照顺序不断地重复，为了区分不同八度的同一分类的音符，通常使用一些辅助标记。在科学音调记号法中，利用分类的字母及表示其所在八度的阿拉伯数字来表示音符。以标准音高 440 赫兹为例，其对应

的音名记号是 A4，表示属于标识为 A 的分类，且在第 4 个八度上。A4 后面的音符为 B4, C4, D4, E4, F4, G4, A5, B5, B6, …。在传统的音乐系统中，一个八度的标记通常由 C 音符开始，也就是说，C4 所在八度的所有音符是 C4, D4, E4, F4, G4, A5, B5。

由于发展的需要，后来在音名后面加上变音记号，如升号或降号。表示在原来音高基础上再升高或降低半音，根据十二平均律（目前最广泛的调音法），即将原频率升高或者降低到 1.0594 倍。升高标记符号为♯，降音符号为♭，如 C♯，D♭表示降 D，可知 B♭与 C♭事实上表示相同音符。这样半音阶在原有的七个音高集合中又增加五个集合，并且任意两相邻的音高集合刚好相差一个半音。对应的 12 个集合在一个八度上的音符如表 2.5 所示，表中任意两相邻音符间相差一个半音，同时表中标记了每个音符对应的 MIDI 编号。任何一个音符与 MIDI 编号都有唯一的对应关系。

不同音符具有不同的频率，相差一个八度的两个音符，音调高的音符频率是音调低的音符频率的两倍。由前述已知，在一个八度中，总共划分了 12 种音高集合，每两相邻集合的对应音符间刚好相差一个半音阶，从而一个八度中有十二个特定频率的音符。这些固定频率之间存在固定的数学关系。一般最基本的音符是 A4，该音符的标准音高一般取 440HZ。

表 2.5　自 C4(中央 C)开始往上八度内的半音阶及对应 MIDI 编号

	主音		第二音		第三音	第四音		第五音		第六音		第七音
自然音	C		D		E	F		G		A		B
升半音		C♯		D♯			F♯		G♯		A♯	
MIDI 音符编号	60	61	62	63	64	65	66	67	68	69	70	71

按惯例，音乐中的音名通常由一个字符、变音记号及用来代表其在哪一八度的阿拉伯数字构成。所有的音符都可用中央 A(A4)的整数倍表示，假设任何一音符与 A4 音符之间的这个整数倍为 n,那么如果一个音符高于 A4 音符，则 n 为正值，否则为负值。由此任何一个音符的对应频率f为：

$$f = 2^{n/12} \times 440HZ \qquad (2\text{-}2)$$

根据表 2.5 的编号规律，不难得出 MIDI 编号与频率之间的关系：

$$f = 2^{(p-69)/12} \times 440HZ \qquad (2\text{-}3)$$

$$p = 69 + 12 \times log_2(\frac{f}{440}) \qquad (2\text{-}4)$$

其中p表示音符对应的 MIDI 编号，这里p可以为小数，f表示音符对应的频率。

由上述的关系表达式可知，MIDI 编号与音符频率，即基频之间存在固定的转化关系。为了更直观的说明这一关系，我们使用一 MIDI 文件边播放边录音，然后对录音提取基频，再将基频通过上述公式转化为 MIDI 音符编号，来验证这一结论。结果如图 2.2 所示，图中第一条曲线为原始的 MIDI 旋律曲线，最后一条曲线为由录音通过提取基频再按上述公式转化得到的 MIDI 旋律。可以看出，从录音提取的 MIDI 旋律与原始 MIDI 旋律保持一致，说明通过提取哼唱语音的基频作为特征并转化为音符是一种合理的方案。

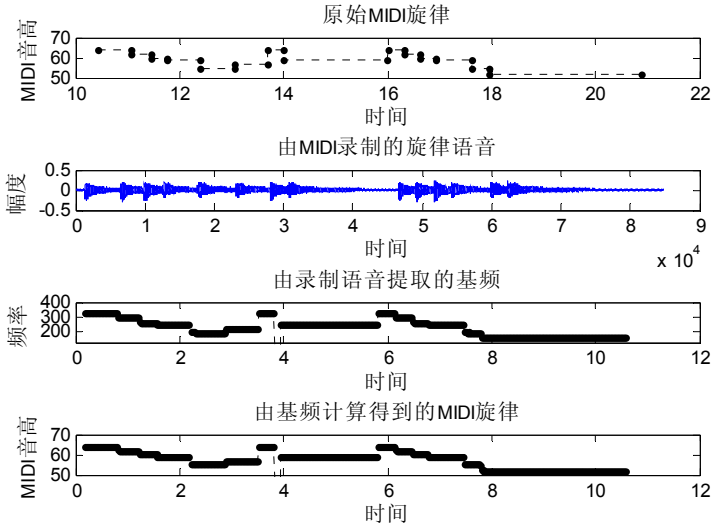

图 2.2　由 MIDI 录制语音并提取基频转化为 MIDI 旋律

2.4　哼唱语音旋律特征的提取

2.4.1　音高的提取

音高提取是语音处理中经常碰到的问题，从上世纪六十年代开始对基频的研究以来，已有很多的研究和改进工作。

尽管没有明确的模型来计算音高，但音高和基频之间存在直接关系，且这种关系表明它们之间可以通过公式相互转化，这一点在上一节中已经证明。因此可以将音高提取转化为基频提取，因此音高提取也称为基频提取或 F0 提取。

基频(F0)是准周期语音信号的最低频率，基频仅仅针对周期语音信号而言。基频和基音周期是相互关联的概念，它们之间有如下关系：

$$T_0 = 1/F_0 \qquad (2\text{-}5)$$

因此，通常将基频提取的问题，归结为基音周期的估计问题。如图 2.3 展示了一段周期语音的基音周期，其中每两个箭头所指位置相差的时间为一个基音周期。

音高提取的关键是对基频或基音周期的准确提取，目前已经有很多的方法来实现，并且大部分算法只要参数选择恰当都可以达到实时计算的要求，因此对于本文的哼唱检索，只需考虑算法特征提取的准确率。一般提取基频的方法有：基于时域的方法、基于频域的方法和基于统计的方法。

（1） **基于时域的方法**。

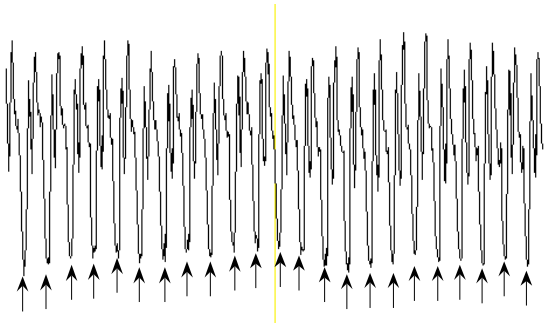

图 2.3　基音周期

这一类方法利用了语音信号的周期性。这种特性使得这些方法遇到非和谐波信号或信号的主要能量集中于高频部分时变得异常脆弱。这里仅以最常用的自相关法为例。

设语音信号为$x(t)$，则其自相关函数定义为：

$$Rx(t) = \int_{-\infty}^{\infty} X^*(\tau)x(t+\tau)d\tau \qquad (2\text{-}6)$$

对于周期信号，其自相关函数的周期与原始信号的周期保持一致[32]。因此，可以利用这一特点来估计语音信号的基音周期。

（2） 基于频域的方法

这里介绍基于倒谱(Cepstrum)方法，语音信号$x(t)$的倒谱定义为：

$$Cx(t) = (\int_{0}^{\infty} log|x(\omega)|^2 cos(\omega t)d\omega)^2 \qquad (2\text{-}7)$$

其中$x(\omega)$表示$x(t)$的傅里叶变换，对于周期信号，同样有信号倒谱的周期性与原始语音信号一致的结论[33]，因此求原始信号的周期，可以转化为求倒谱信号的周期。

在实际提取过程中，需要对原始语音信号进行加窗处理，由于这种方法和（1）中的方法都是用于估计原始信号的基音周期，要求窗长不能太短，所加窗内至少包含两个基音周期，也不能太长，如果窗长太长，则窗内包含了多种基音周期，对基音周期的估计会出现偏差。对于人在哼唱中的基频，与正常语音的基频范围大致相同，一般男声的最低基频可达到 40 赫兹，而女声总体比男声高，女声最高基频可达到 600 赫兹。在对哼唱语音提取基频时，应当根据人发声的基频范围，设置合适的窗长提取。

（3） 基于统计的方法

这类方法利用的是具有相同音高的语音帧之间的相似度。对语音中每一帧，通过提取和基频有关的特征，再通过统计的方法进行分类，根据在基频上的差异分成不同类别。这些方法的弱点是必须使用训练数据。因此，训练数据的质量及训练过程都会影响最终模型输出结果时的准确率。

目前这类方法主要有基于神经网络[34, 35]和 HMM[36, 37]的方法。以 HMM 的方法为例，从统计学观点来看，一段哼唱语音信号，可以看成是一个未知过程

输出的观察序列，而这个未知过程正是哼唱者想要哼唱的内容，但所能观察到的只有语音信号，及提取的语音特征。因此，这一过程可以使用 HMM 进行模拟。HMM 中的转移概率需要考虑给定音符自身的过程包括音符开始发声、持续及停止的过程，还要考虑音符转移到其他音符的过程。一个音符完整的发声过程可以认为由开始发声（由其他音符转移到当前音符，音高逐渐变为音符的标准音高）、持续（音高保持不变，并持续一定时间）和停止（音高逐渐变化开始为哼唱下一个音符做准备）三种状态组成。HMM 中的输出概率可以认为是某一音符输出对应的语音信号特征的概率。这样音高提取过程就变成搜索音符内及音符间迁移使这一过程输出对应的语音信号特征的概率达到最大的过程，这一搜索过程可以使用 Viterbi 解码来解决。

定义了音符内的状态迁移及音符间的迁移，便可以构造一个两层的 HMM 结构。这点类似于使用 HMM 进行汉语语音识别的过程，音符类比于汉语中的词，音符内的状态类比于汉语中的声韵母。假设我们定义音符之间的迁移所在的层次为事件层次，而音符内的迁移为发声层次。在事件层次上的每一状态对应了不同的基频，哼唱的旋律被描述成这些状态构成的序列。如图 2.4 中（a）所示为事件层次上不同状态之间的迁移过程，每个状态迁移到其他状态的概率和状态的持续时间有关。这可以通过对旋律数据进行统计得到。而在发声层次中的每一状态表示了音符发声的某一阶段，如图 2.4 中的（b）所示，图中 a 表示开始发声，s 表示持续，r 表示声音停止，其中一个音符可以不需要停止就直接由持续状态迁移到另一音符的发声状态。

定义了 HMM 中的状态，对应的语音模型便可以通过训练数据得到，然后通过 Viterbi 解码过程得到语音信号中的音高，这种方式同时提取了音符信息。

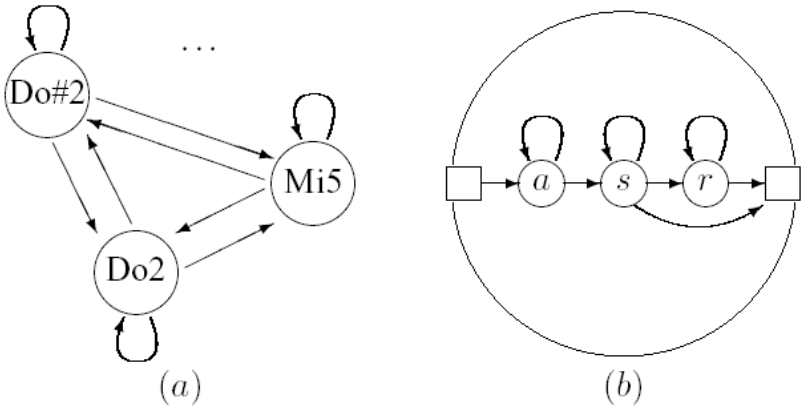

图 2.4　HMM 中两个层次的状态转移

2.4.2　音符的提取

音符提取方法大致可分为非统计和统计的方法。

（1）基于非统计的方法

对于基于非统计的方法，关键问题是音符起点的估计问题，估计出每个音符的起点后，便可结合语音信号中的静音、音高等其他特征进行音符分割，从而给出语音信号的音符序列。

如图 2.5 为音符起点(onset)的示意图，一般在哼唱的音符开始时，语音信号能量会突然上升（attack），然后持续一段瞬时时间（transient），再逐渐下降（decay），这段瞬时时间的起始点，称为音符起点。

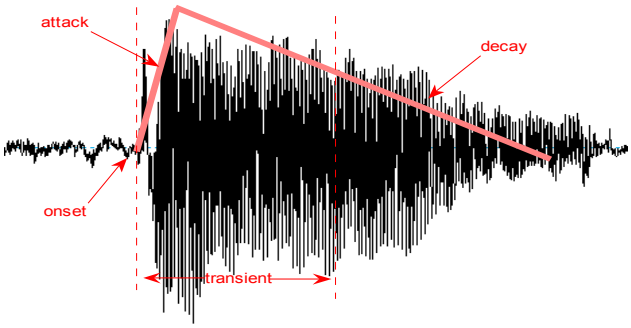

图 2.5　音符起点

音符起点检测是音符提取中一个重要的步骤，通过音符起点检测，将每一个音符从语音信号中分割出来，并得到每一个音符的时长信息。再结合预先提取的音高信息，计算每个音符的音高，便将一段旋律表示成了具有音高和时长的音符组成的序列。经过这种处理的特征，和 MIDI 旋律的音符特征保持一致，便可以和 MIDI 旋律进行匹配了。

在哼唱检索系统中，音符起点检测、音符分割的好坏，直接影响匹配环节，对整个系统的准确率有很大影响。因此，对于基于非统计方法进行音符提取的哼唱系统，音符起点检测在系统性能中扮演了重要角色。

对于音符起点检测的相关研究中[38, 39]，一般将音符起点检测分成预处理、简化、峰值提取及决策四个阶段。语音信号经过预处理，将原始语音信号转换到频域用于分析或进行其他特征计算。然后通过简化处理，从转换后的信号中提取能够表现音符起点的特征序列如能量，一般在音符起点之后，语音信号的能量会出现突然上升的趋势，因此这种能量突然上升可作为音符起点检测的依据，这一简化过程也被称为检测函数。经过检测函数处理后的信号，在音符起点部分与其他部分有显著的区别。接着进行峰值点提取，峰值点提取的主要目的是从检测函数的输出结果中，选取可能为音符起点的特征点，作为音符起点候选。如当检测函数对音符起点的能量突变有较大输出值时，根据峰值提取得到的大部分候选时间点，即是音符起点。最后进行决策，决策的过程是通过一定的判断函数，来决定由检测函数给出的峰值点是否是音符起点，最后给出所有音符起点。

（2）基于统计的方法

基于统计的方法较为常用的是基于 HMM 的方法，已知语音信号的观察序列为 $X = X_1, X_2, \cdots X_n$，则语音识别的目的是找出这样一个词序列 $\widehat{W} = w_1, w_2, \cdots, w_m$，使得该词序列对观察序列具有最大的后验输出概率。最大后验输出概率定义为：

$$\widehat{W} = \arg\max_{W} \frac{P(W)P(X|W)}{P(X)} \tag{2-8}$$

在哼唱检索中，音符提取同样可看成是语音识别的过程，上述的W可以看成旋律中的音符序列，X可以看作音符序列对应的观察特征向量。然后训练一个语音模型用于计算输出概率$P(X|W)$，同时训练一个语言模型计算先验概率

$P(W)$。最后通过一种“one global search process”[40]的方法来进行语音到音符间的转换。

通常音符提取是以音高序列为基础再通过对音高序列进行分割归并得到音符的过程，每一个音符由一段对应的音高序列转化得到。尽管这种方式比较直观，在语音与非语音之间给出明确的界限却比较困难。即便是最好的音高提取算法，也不免会有分类错误的情况存在，这些错误对算法提取音符造成不利影响，如果语音带噪情况比较明显时这种影响就更严重。为了提高语音模型的鲁棒性，在 HMM 中，可以使用高维的倒谱特征作为音符的表达形式，而不再使用音高序列来表示[36]。

如图 2.6 所示为音符 C4（对应 MIDI 编号为 60，基频为 220HZ）的某一语音帧的倒谱（上）及一非语音帧的倒谱（下）。倒谱图（a）中的低频部分由于不可能包含任何音符信息，因此被截去。可以看出，对于语音帧而言，其倒谱图中有多个峰值点，而对于非语音则不存在明显的峰值点。在倒谱中，横轴参数x与对应频率及采样率之间有如下的关系：

$$x \times f(x) = sampleRate \tag{2-9}$$

其中$f(x)$表示x对应的基频。已知语音帧的基频为 220HZ, 并且已知语音的采样率为 16kHZ, 按此公式计算得到的峰值位置应当在$x = 16000/220 = 72$，与图中（a）对应倒谱的第一峰值点符合。

倒谱特征再经过一定的预处理便可用于训练语音模型。首先，从倒谱中取出与基频相关的区域特征，这个区域的下限值对应人可能的最高基频，上限值对应人可能的最低频率。对男性的哼唱语音而言，在采样率为 16kHZ 的条件下，可取$x = 48$到$x = 240$之间的区域，这正好对应于音符 C2 到 E4（对应的 MIDI 编号为 36 到 64）之间的音的频率范围，转化为对应的基频则为 66.6HZ~333.3HZ，符合男性的基频范围。对于女性哼唱，同样在 16kHZ 的条件下，可取$x = 24$到$x = 120$的区域用于提取特征，这对应于音符 C3 到 E5（对应 MIDI 编号为 48-76）之间的音符发音，表示的基频范围为 133.3HZ 到 666.6HZ，也符合女性基频范围。确定倒谱中的区域后，对应区域被压缩成固定长度，并根据压缩比率对倒谱中的特征值进行适当调整。由于在以上过程中，男性的区域宽度是女性的区域宽度的两倍，被压缩后就相当于将男性的音高提高了一个八度。

倒谱中的区域经过压缩，归一化到同样长度，便可以提取倒谱特征训练语音模型了。

在 HMM 中，除了语音模型，还需训练语言模型。如前文所述，在西方音乐记号系统中，一个八度内有 7 个主音，包含 12 个半音音符。即有些半音音符可能在一首歌的所有旋律中都不出现。因此，在基于 HMM 的音符提取系统中加入语言模型有利于系统正确率的提高。有两种方法可以用来训练语言模型，一种是音符相关的，另一种是主符无关的，对于前一种，需要对每一个主音建立一个语言模型并且语言模型是跟哼唱信号相关的。如果语言模型使用不当，反而会使结果变差。而对于音符无关的方式，用于训练的旋律中的音符被调整到其他的 11 种音符，这样原始的旋律与被调整的旋律构成一个更大的数据库来训练 n-gram 的语言模型，这里的 n 一般取值为 4。

训练得到语音模型和语言模型后，即可按照 HMM 的方式，对任何一输入的哼唱语音，通过提取每一语音帧的特征构成观察向量，得到最终提取的音符序列。

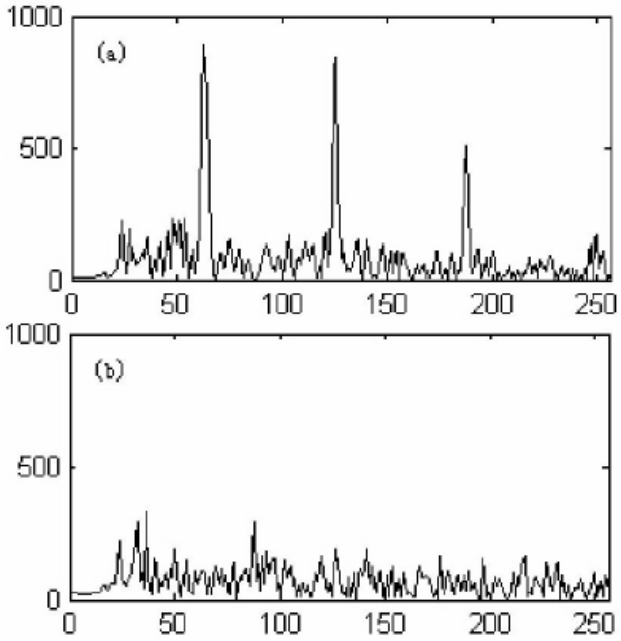

图 2.6　(a)为音符为 C4 的语音帧的倒谱，(b)为非语音帧的倒谱

2.5　哼唱旋律与 MIDI 旋律的匹配

2.5.1　匹配的定义

哼唱检索中的匹配，是指根据用户以歌词唱、哼唱或者吹口哨、打拍子等形式录制的哼唱语音，提取一定的特征，并和数据库中预先存储的特征计算相关度。

2.5.2　匹配的特征

用于匹配的特征根据不同匹配算法可以使用多种形式的特征，如基音周期、节拍、停顿、音高的升降等，一般常用的特征是哼唱语音的基音周期（Pitch），基音周期的提取可以使用自相关法[41]、循环 AMDF 方法[42]、基于 CEP 和 LPC 谱的方法[43]、小波分析[44]等，通过提取得到基音周期的序列，转化成对应的 MIDI 音高，再对音高进行归并，得到 MIDI 音符序列，至此即可使用音高序列和音符序列作为特征来进行检索匹配。

本文使用的提取基音周期的方法是基于 CEP 的方法，提取音符序列则使用 HMM 的方法[36]，而数据库使用 MIDI 数据库。获得基音周期后，可以计算得到对应的基频，基频和 MIDI 的音符之间存在对应的转化关系，在使用 MIDI 标准时，MIDI 音符的音高和频率之间的对应关系为：

$$p = 69 + 12 \times log_2(\frac{f}{440}) \qquad (2\text{-}10)$$

其中f表示频率，p表示 MIDI 音符的音高，440 是的标准音高（440 赫兹）。由此可以看出，比较哼唱查询语音和 MIDI 旋律之间的差异，实质上是比较它们之间在频率变化上的差异。

2.5.3　匹配的难点

通过将每帧基频转化为音高，构成音高的序列，同时再使用 HMM 的方法对音高序列归并提取音符序列，再与数据库中 MIDI 音符序列进行比对。如图 2.7 所示，第一条曲线为查询语音的音高序列曲线，第二条为查询语音的音符序列曲线，第三条为数据库中对应的目标 MIDI 的音符序列曲线。第二条曲线中的非水平部分为哼唱查询语音中的非语音部分。可以看出，MIDI 音符序列比较标准，音符长度的变化较为平缓，而查询语音的音符序列则不规则，但是总体

变化趋势和目标 MIDI 音符序列基本一致。同时哼唱语音的速度和 MIDI 旋律的速度有一定缩放，总体的音符音高和 MIDI 音符之间也存在一定的纵向偏移。

由此可以看出，对于匹配算法，必须考虑如下情形：

（1）音高偏移：哼唱查询语音的音符与实际的目标旋律音符存在一定差异，即音调总体偏高或偏低；

（2）线性伸缩：哼唱查询语音的速度和目标 MIDI 旋律的速度存在一定差异，或快或慢或快慢不均；

（3）音符不匹配：尽管哼唱查询语音与目标 MIDI 的旋律在大体轮廓一致，哼唱语音中的音符相比目标 MIDI 中的对应音符，或长或短，或缺失或被分成 2 个音符等。

对于如上所述的 3 种情形是研究匹配算法需考虑的，也是匹配算法要解决的难点。前 2 条针对两段旋律间的整体情况而言，第 3 条是针对两段旋律间每个音符之间匹配的情况。

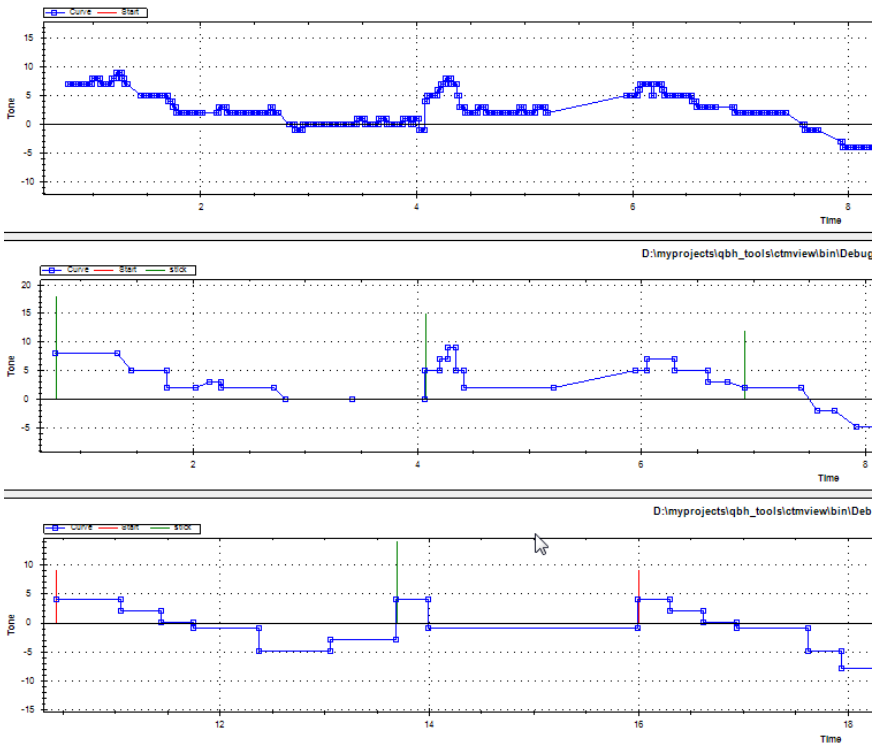

图 2.7　哼唱查询语音的音高、音符序列及目标 MIDI 的音符序列片段

2.5.4　匹配的分类

根据匹配算法的层次，可以将匹配算法分成快速匹配和精确匹配。

快速匹配的目的是对系统由 MIDI 数据库生成的大量候选，利用一定的算法，快速从大量候选中去除可能性较小的候选，减少后期匹配的计算量，提高系统时间响应性能。快速匹配强调匹配速度，同时也强调在一定程度内保持正确的候选。[27]中就使用了多种不同的特征和不同的方法来进行快速匹配，如使用前几个音符、方差、最大音高等特征，并用线性分类器及简化版的递归方法进行快速匹配以保证计算速度。

精确匹配的目的是将哼唱旋律片段和所有候选旋律片段进行更精确的相似度计算以获取最终候选列表。精确匹配的特点是计算量大、时间消耗大但计算结果更精确。精确匹配一般处于系统后端，其计算结果直接输出作为最终候选。如常用的 DTW 算法在精确匹配中使用较多，DTW 方法能较好地解决上述提到的匹配难点中的后 2 条，而对于音高偏移，则难于解决，必须依靠其他方法估计音高偏移，或通过设置不同音高偏移计算 DTW 来估计最佳音高偏移同时获得最小差异值。如[3]中使用启发式方法来估计最佳音高偏移，通过以当前音高偏移值附近的其他值来计算 DTW 分数并和当前音高偏移对应的 DTW 分数比较，不断调整当前音高偏移值直到 DTW 分数不再下降，来计算最佳的匹配分数。

2.6　哼唱旋律与 MIDI 旋律的匹配算法

2.6.1　基于字符串匹配的方法

字符串匹配方法源于对旋律变化的直观感觉，是对旋律变化信息作一定的简化处理进行的匹配。通过将旋律提取为音符序列，再根据两相邻音符之间的升高、降低或音高不变的情况将旋律表示为特定的字符串。当然，也可以使用其他的方式将旋律转化为字符串，这里以最简单的即音符的升降为例来介绍。

在获取旋律的音符序列后，为描述旋律的变化信息，字符串匹配方法将旋律的变化根据上升、下降和不变，分别标记成 U/D/S 三个字符，由此构成一个由 U/D/S 三个字符组成的字符串序列。如图 2.8 所示为一段示例旋律的字符串表示，该旋律对应字符串为 UDDDDDUSUUDD。

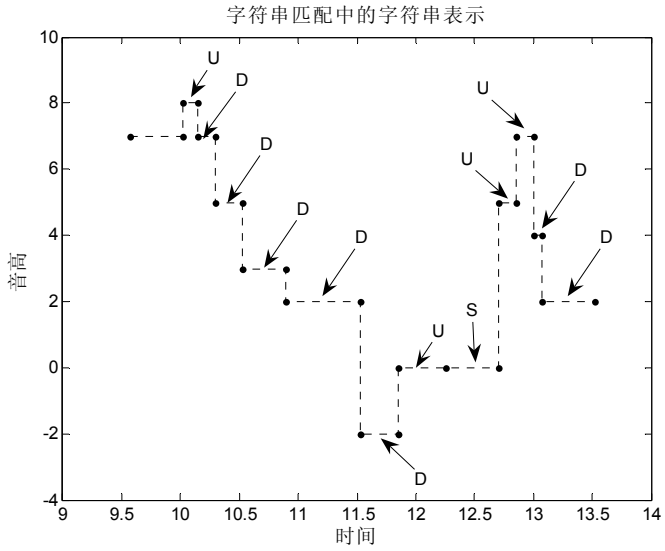

图 2.8　字符串匹配中音符变化的字符串表示

在将数据库旋律及哼唱旋律都转化为字符串之后，使用字符串匹配算法计算二者的距离，由于音符提取时会存在一定的错误，要求字符串匹配算法必须容忍一定的错误，由于具体算法与本文研究课题无关，可参考相关文献[12, 14]。

2.6.2　线性伸缩

线性伸缩方法通过将其中一旋律进行拉伸及音高平移，达到与另一旋律对齐匹配的目的。由于不同人的音域不同，导致哼唱同一旋律时，音调或比数据库标准旋律高，或低，但变化基本一致，因而在人听来依然是同一歌曲。同理，不同人对速度的把握也不一致，因而也存在不同人的哼唱旋律与数据库旋律存在不同速度比的情形。在拉伸系数及音高偏移得到有效估计的条件下，线性伸缩方法能够有效地检索出目标旋律。

如图 2.9 所示为线性伸缩的示例，这里没有标记音高偏移，哼唱旋律经过不同拉伸系数进行拉伸后，再经过纵向平移（即音高整体加一个偏移值），然后与数据库 MIDI 旋律进行对齐计算相似度。

线性伸缩方法计算起来简单，同时较直观，在旋律起点对齐正确的情况下，只要找到正确的拉伸系数和音高偏移，就能找到目标旋律。其缺点是：1. 对起点对齐的要求较高，如果对齐出现错误，则哼唱旋律无论如何拉伸平移，都很难与目标旋律达到理想的匹配结果。如果对数据库旋律中的每一音符都作为起

点与哼唱旋律进行线性伸缩匹配并估计伸缩参数，则计算量太大。 2. 必须选择适当的拉伸系数和音高偏移。哼唱旋律和 MIDI 旋律一般不适合直接进行匹配，必须预先估计拉伸系数和音高偏移，尽管可以使用穷举法计算最佳参数，但计算量太大。因而需要在不增加过多计算量的前提下估计出合适的拉伸系数和音高偏移；3. 当哼唱旋律长度较大时，使用同一的拉伸系数和音高偏移效果不再明显。使用线性伸缩进行匹配的理想条件是哼唱旋律在各处的音符与目标 MIDI 旋律中的对应音符具有一致的音高偏移和拉伸系数。但这一条件在实际中很难达到，从图 2.3 可以看到，当哼唱旋律经过拉伸与 MIDI 旋律局部对齐较好时，其他部分的对齐则可能较差，很难达到各处均达到最佳对齐的效果。

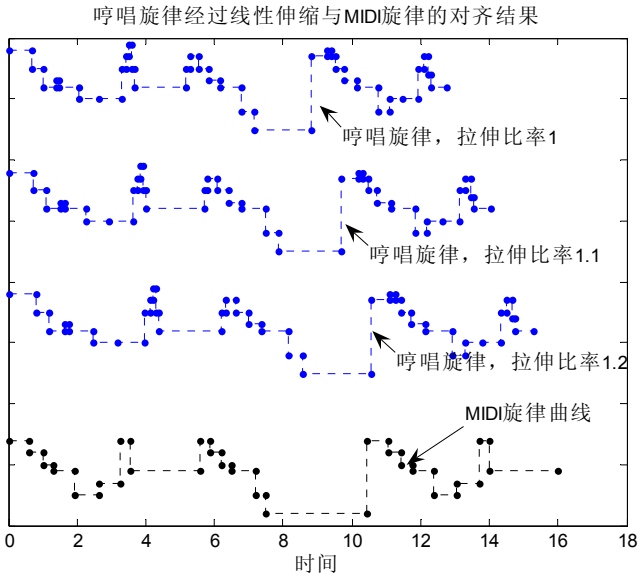

图 2.9　线性伸缩进行匹配的示例：哼唱旋律经 1 倍、1.1 倍及 1.2 倍拉伸后与 MIDI 旋律对齐匹配

2.6.3　动态规划

动态规划应用于哼唱检索，在实现时必须施加一定的限制，受特征提取等方面的影响，可能会出现 MIDI 旋律中的一个音符对应于哼唱旋律中的几个音符的情况，但无论如何，该音符的时间和对应的几个音符的累积时间之间的比率应当不会超过某一比率值。

假设哼唱旋律的音符序列为$Q_m = x_1, x_2, \cdots, x_m$，其中$x_i = (p_i^q, d_i^q)$，$x_i$表示第$i$个音符，$p_i^q$表示第$i$个音符的音高，$d_i^q$表示第$i$个音符的时长。MIDI 旋律的音

符序列为$M_n = y_1, y_2, \cdots, y_n$，其中$y_i = (p_i^m, d_i^m)$，$y_i$表示第$i$个音符，$p_i^m$表示第$i$个音符的音高，$d_i^m$表示第$i$个音符的时长。同时假设哼唱旋律相对 MIDI 旋律的音高偏移为k，假设在动态规划过程中会有一个音符匹配多个音符的情况，且无论是 MIDI 音符匹配哼唱旋律中的多个音符，还是哼唱旋律中的一个音符匹配 MIDI 旋律中的多个音符，均限制 MIDI 旋律的累计时间与哼唱旋律的累计时间之间的比值必须在区间$[minR, maxR]$内。则Q_m与M_n的动态规划距离$DP(Q_m, M_n)$可描述为：

$$\begin{cases} min(DP(Q_{m-1}, M_{n-k}) + LS(k, [x_m], [y_{n-u+1}, \cdots, y_n])) & if\ p_m^q > p_n^m \\ min(DP(Q_{m-k}, M_{n-1}) + LS(k, [x_{n-v+1}, \cdots, x_m], [y_n])) & else \end{cases} \tag{2-11}$$

其中u, v满足：

$$\begin{cases} minR \leq \dfrac{\sum\limits_{n-u+1 \leq t \leq n} d_t^m}{d_m^q} \leq maxR \\ minR \leq \dfrac{y_n}{\sum\limits_{n-v+1 \leq t \leq m} d_t^q} \leq maxR \end{cases} \tag{2-12}$$

其中$DP(Q_m, M_n)$表示使用动态规划方法计算Q_m和M_n之间的差异值。$LS(k, [x_m], [y_{n-k+1}, \cdots, y_n])$表示使用线性伸缩方法并且设置音高偏移系数为$k$计算音符序列[$x_m$]和[$y_{n-k+1}, \cdots, y_n$]之间的差异值。$LS(k, [x_{n-k+1}, \cdots, x_m], [y_n])$则表示使用线性伸缩方法并设置音高偏移系数为$k$计算音符序列[$x_{n-k+1}, \cdots, x_m$]与[$y_n$]之间的差异值。

动态规划的问题，实际上是一个递归最优的问题，并且在递归的过程中需要使用线性伸缩匹配。这里的线性伸缩匹配两段旋律的左右边界已经确定，并且音高偏移已经预先提供，因而线性伸缩较为容易计算。同时，这种动态规划方法将线性伸缩细化到了不能再细化的地步，对匹配中的每个音符给予不同拉伸系数。相对而言比直接进行线性伸缩匹配更为灵活，当然，这种方法的计算量也比较大。

2.6.4　基于递归的线性伸缩方法

这种方法采用递归的按中点切分进行线性伸缩匹配来探索最优的匹配结果[27]。由于这种方法属于分段匹配的方法，与本文研究内容关系较密切，因此做详细介绍。这种称为递归对齐(Recursive Alignment, RA)的方法从线性伸缩匹配

扩展而来但是比线性伸缩方法更好。它克服了线性伸缩在匹配时对旋律总体使用单一的拉伸系数和音高偏移而导致的匹配不够精确的问题。同时，这种方法既满足了线性伸缩的要求，又是一种自顶向下、由粗而细的方法，对旋律粗轮廓的匹配优于直接动态规划的方法。

根据原文，该种算法中 MIDI 旋律使用音符表示，而哼唱旋律使用的是音高序列。音高序列即未进行归并得到音符前的序列，由每一帧的音高构成。设哼唱旋律$Q=(q_1,q_2,\cdots,q_n)$，其中q_i表示第i帧的音高，n表示音高序列的总长度。设 MIDI 旋律为$N=((p_1,d_1),(p_2,d_2),\cdots,(p_m,d_m))$，其中$(p_i,d_i)$表示第$i$个音符的音高和时长，这里的时长以帧数表示，与哼唱旋律相对应。设$S=((sx_1,sy_1),(sx_2,sy_2),\cdots,(sx_n,sy_n))$为预先设定的拉伸系数和音高偏移的可选参数列表。再设D为最大可递归的深度，一般设置为 3 或 4。则算法描述如下：

$function$ **RA**$(\mathbf{Q},\mathbf{N},\mathbf{S},\mathbf{D})$:
    $i\leftarrow 0, j\leftarrow \lfloor m/2\rfloor$, $maxScore\leftarrow -\infty$
    $sc\leftarrow \sum_{l=1}^{j} d_l / \sum_{l=1}^{m} d_l$
    $N_1\leftarrow((p_1,d_1),\cdots,(p_j,d_j)), N_2\leftarrow((p_j,d_j),\cdots,(p_m,d_m))$
    for $all\ (sx,sy)\ in\ pair\ set\ S$ **do**
        $k\leftarrow\lfloor sx\cdot sc\cdot n\rfloor$
        $Q_1\leftarrow(p_1,\cdots,p_k), Q_2\leftarrow(p_k,\cdots,p_n)$
        $score\leftarrow LS(Q_1,N_1)+LS(Q_2,N_2)$
        $if\ score > maxScore\ then$
            $maxScore\leftarrow score$
            $i\leftarrow k$
        end if
    end for
    if $D=0$ **then**
        return $maxScore$
    else
        $Q_1\leftarrow(p_1,\cdots,p_i), Q_2\leftarrow(p_i,\cdots,p_n)$
        return $RA(Q_1,N_1,S,D-1)+RA(Q_2,N_2,S,D-1)$
    end if

上述算法中的$LS(Q_1,N_1)$表示使用线性伸缩的方法对音高序列Q_1和音符序列N_1计算差异值，$LS(Q_2,N_2)$亦同理。$RA(Q_1,N_1,S,D-1)$则表示对子序列Q_1和N_1继续使用参数S再做$D-1$层的递归调用。

该算法实际上是通过不断的分段调用线性伸缩方法计算差异值，同时递归调用获取最佳分段方案及最小差异值。这种方法的弊端也是很明显的，线性伸缩的拉伸系数及音高偏移均是预先设定的，由于拉伸系数和音高偏移的组合比较多，如果S中的组合数太少，则对准确率有一定的影响，而如果S中的组合数太多，则会增加不必要的计算量。

2.7　本章小结

本章从哼唱检索的基本原理出发，介绍了 MIDI 的数据格式，从而引出音符与基频之间的关系，接着分析了音高和音符特征的提取，最后阐述了旋律匹配的相关概念，并详细介绍和分析了最早的及跟本文研究关系较为密切的几种方法，分析了这些方法的优缺点。当然，实际上还存在其他的一些算法，但是由于与本文的研究关系不大，因此不做详细介绍。

第3章　分段信息

3.1　本章引论

在第 2 章中描述了匹配的难点，哼唱查询语音的旋律曲线与目标 MIDI 旋律曲线之间，尽管大致轮廓及变化趋势保持一致, 却并非完全一致。使用线性伸缩方法进行旋律匹配，首先面临的问题是旋律归一化的问题，即在旋律进行对齐之前需要先对哼唱旋律和 MIDI 旋律进行归一化，以使二者尽量保持相同的速度和音高，尽管启发式估计算法有利于找到最优的拉伸系数和音高偏移两个参数，但可能陷入局部最优解。另一个问题是线性伸缩的全局参数问题，对于音高偏移，由于哼唱者的个人音色不同，不同人与同一 MIDI 旋律之间有不同的音高偏移，同时由于个人水平、对歌曲的熟悉度等原因，同一人哼唱同一首歌的不同旋律片段时也很难保持与 MIDI 完全一致的音高偏移，因而音高偏移是随人随时间不同而变化的。对于拉伸系数，也有相同情形，哼唱者很难做到哼唱的每一片段都与 MIDI 的对应旋律在速度上保持完全一致的速度比，同时在哼唱句子的切换过程中最容易导致拉伸系数的改变。从另一个角度来看，哼唱旋律与目标 MIDI 旋律的线性伸缩参数在局部保持稳定，只是由于在特定时间点由于换气等原因导致变化而影响了参数。

针对旋律归一化以及匹配时局部参数不一致的问题，本文提出了两种特征来解决这两个问题，一种是基于极值点的分段信息，由于其对齐的效果比较好，用于增加启发式估计算法的候选起点。另一种是基于停顿点的分段信息，由于这种信息可以合理的将哼唱旋律分成不同句子，相当于乐谱中的小节，可以利用这种信息进行分段地线性伸缩匹配。

本章主要阐述极值点和停顿点两种旋律特征点，主要介绍它们的定义，优点以及估计方法等。

3.2　基于极值点的分段信息

3.2.1　极值点的定义

极值点即在旋律音符曲线中升到最大值再转而下降的点，或者是降低到最小时转而升高的点，本文主要讨论最高点，如无特别说明，以下所说的极值点都指最高点。本文所说的旋律曲线的极值点和一般的极值点定义稍有不同。旋律曲线中的最高点（最低点），只有当其相对两侧“升高”（“降低”）的幅度超过指定阈值, 这样的最高点（最低点）才称为极值点。这里升高或降低幅度的计算方法也和一般的计算方法不同，在计算最高点对左侧上升的幅度时，从该最高点往前追溯，并跳过伪最高点和伪最低点，直到追溯到一个最低点，则该最高点和追溯到的最低点之间的音高差即为该最高点左侧相对升高的幅度值，最高点对右侧升高的幅度也通过类似方法计算。如图 3.1 所示为一最高点及两侧追溯到的最低点的示例。

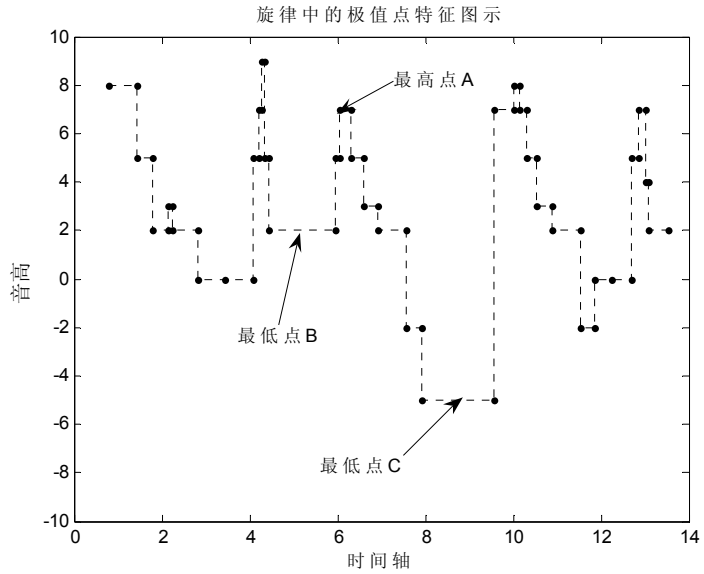

图 3.1　旋律中极值点特征图示

极值点中的伪最高点和伪最低点，可以认为是旋律的升高或降低等过程中因干扰或旋律本身变化不大的凹凸变化，由于这种小的凹凸变化至少对一侧相对升高的幅度太小，认为是伪最高点或伪最低点。以伪最高点为例，图 3.2 列出了三种类型的伪最高点，第一种为旋律上升过程中的伪最高点，第二种为旋

律下降过程中的伪最高点，第三种则为旋律保持平稳过程中的伪最高点。由于伪最高点的变化较小，特征不明显，极有可能是因为哼唱过程中的不稳定或噪声等因素导致的短时起伏，因此在提取极值点过程中需将此类极值点作为伪极值点去除。

旋律中的极值点代表了旋律中较高亢的部分，同时预示了旋律音高局部变化的边界值，因此是重要的参考点。假如一段旋律由最低点升高到最高点再降低到最低点，那么可以认为另一段旋律也应当做类似的升降变化，在听觉上才认为相似。如图 3.3 所示为歌曲“冬季到台北来看雨”的前 3 句哼唱以及对应 MIDI 旋律的最高点标注。通过将图中哼唱的语音信号与旋律的极值点标注可以看出，极值点的位置对应的语音信号能量也比较强，这就印证了极值点是旋律中高亢部分的说法。另外，图中哼唱旋律的第一个最高点和第二个最高点之间跳过了一个伪最高点。从图中还可看出，通过对旋律做最高点分割后，每两段对应的片段在变化上极相似。若通过此种方式分段后，利用这种分段信息向启发式线性伸缩参数估计算法增加候选起点来估计线性伸缩参数，将有利于对线性伸缩参数的估计。

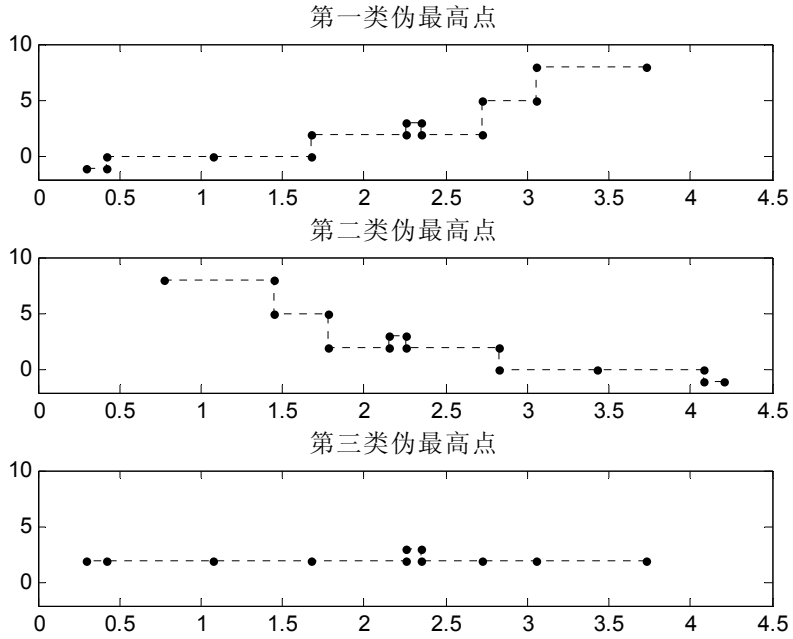

图 3.2　伪最高点的三种类型

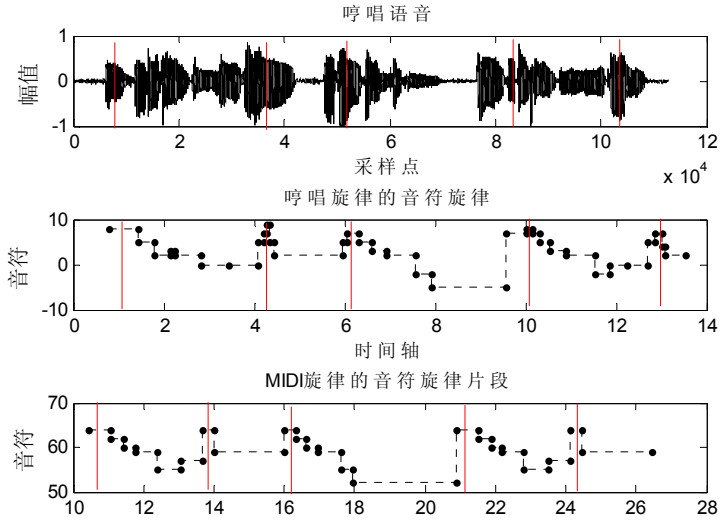

图 3.3　哼唱旋律和 MIDI 旋律的最高点

3.2.2　极值点的估计

对极值点估计的思路是：先从旋律中找到比左右音符音高大的最高点，然后对最高点的左右两侧分别计算相对升高的幅度，根据升高的幅度是否满足预先设定的阈值，决定是否将最高点作为伪最高点舍弃。

估计算法思路如下：

对旋律中每一个比左右两侧高的音符：

（1）对左侧向前追溯，跳过伪最高点，计算相对升高的幅度

（2）对右侧向后追溯，跳过伪最高点，计算相对升高的幅度

（3）如果有任何一侧的下降幅度小于设定值，则当前最高点作为伪最高点去除，否则加入极值点列表。

具体算法描述如下：

将旋律中每一音符的音高构成音高序列，假设为：$T = t_1, t_2, \cdots, t_n$，其中$n$表示音符个数。

（1）对T从左往右直到找到位置k，满足：

$$t_{k-1} < t_k > t_{k+1}$$

（2）令索引位置$idx = k - 1$, $acc = 0$, $lastIndex = j - 1$，当$idx > 0$时重复：

$$idx = idx - 1$$

若$t_{idx} > t_{idx+1}$，则：

$acc = acc + t_{idx} - t_{idx+1}$

如果$acc > minJump$, 转 3。

否则：

$acc = 0, lastIndex = idx$

（3）若$t_k - t_{lastIndex} < minDrop$，转 1 继续。

（4）令索引位置$idx = k + 1, acc = 0, lastIndex = j + 1$，当$idx < n - 1$时重复：

$idx = idx + 1$

若$t_{idx} > t_{idx-1}$，则：

$acc = acc + t_{idx} - t_{idx-1}$

如果$acc > minJump$，转 5。

否则：

$acc = 0, lastIndex = idx$

（5）若$t_k - t_{lastIndex} < minDrop$，转 1 继续。

（6）将位置k加入最高点的位置列表，若$k < n - 1$转（1）继续，否则算法结束，返回位置列表。

上面的算法中用到了 2 个参数，$minJump$，$minDrop$，这两个参数均为预先设定的阈值，具体含义解释如下：

指针最高点往两侧移动以估计其升高的幅度，如果碰到一段音高上升的旋律，考虑可能碰到伪最高点，指针继续移动，直到旋律连续上升超过了$minJump$值，此时指针停止移动，并将最后一次下降到的音高值作为估计的最低点的值，和最高点计算音高差值，作为升高的幅度，只有当最高点相对两侧所升高的幅度值均超过$minDrop$时，此最高点才被确认为最高点，否则作为伪最高点被舍弃。

3.3　基于停顿点的分段信息

3.3.1　停顿点的定义

停顿点是指在旋律哼唱过程中停顿的位置，所谓停顿，是指发声过程的停止并保持一段时间。由于这种声音的停止，造成在语音信号中存在短暂的非语音信号并持续一定时间。因此，停顿点实际上表现为一个非语音片段。如图 3.4 所示为停顿点的示例，图中的哼唱语音总共哼唱了 5 句歌词，共出现了 4 个较

长的非语音片段，即该哼唱语音中包含 4 个停顿点。

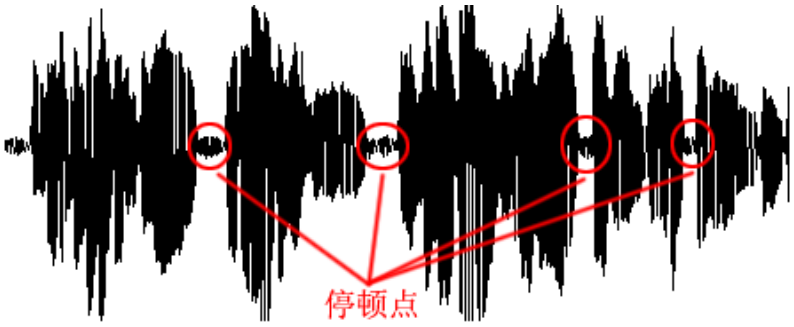

图 3.4　停顿点示例

停顿点的产生是基于旋律节奏的需要，在乐谱中每一段旋律都由小节组成，在人哼唱旋律的过程中，需要在不同小节的切换过程中换气，或停顿，以保证下一小节的哼唱及旋律的节奏感，从而导致哼唱的语音信号中出现持续一定长度的非语音信号。

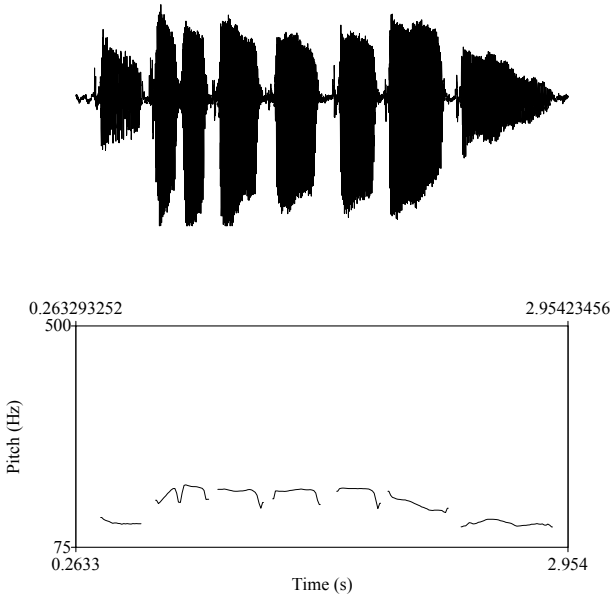

图 3.5　语音信号中的停顿以及在音高提取的表现

如图 3.5 所示，由于人在换气或者停顿的过程中，对应的语音信号中就不存在包含人声的语音流，经音高提取，对应的音高旋律曲线中会出现一定的“空缺”片段。当音高旋律经过特征提取算法分割成音符序列时，这部分“空缺”

片段被归并成一个单独的音符，称为静音音符。这种持续一定时间的静音音符，即为本文中所说的停顿点。

3.3.2　停顿点的物理意义

停顿点是由于旋律中小节的切换产生的，休止符作为小节的结束标志，是音乐乐谱中表示乐音休止的符号。在一首特定的旋律中，每个休止符具有固定的时间长度，长度大小与旋律速度有关。根据休止时间的不同把休止符分成不同类别，相邻的两类休止符之间的时间长度是两倍的关系。

表 3.1　乐谱中的休止符记号

五线谱记谱	简谱记法	休止符名称
	0 0 0 0	全休止符
	0 0	二分休止符
	0	四分休止符
	0	八分休止符
	0	十六分休止符
	0	三十二分休止符
	0	六十四分休止符

如表 3.1 所示为乐谱中的不同休止符记号，在旋律的速度固定的情况下，按照停顿的长度，定义了不同的休止符。全休止符的时间长度计算公式是：

$$T = \frac{60}{S} \times N \qquad (3\text{-}1)$$

其中S表示旋律速度，即每分钟的拍数，N表示以几分音符为一拍。如在一首每分钟 120 拍，4 分音符为一拍的旋律中，一个全休止符的时间长度是 2 秒，对应的二分休止符时间长度是 1 秒。

在乐谱中，休止符的时间长度根据其为几分休止符与对应的乐音音符的时间长度保持一致，每个乐音音符都有对应的休止符，如图 3.6 所示。休止符的音高与对应音符的音高一致。

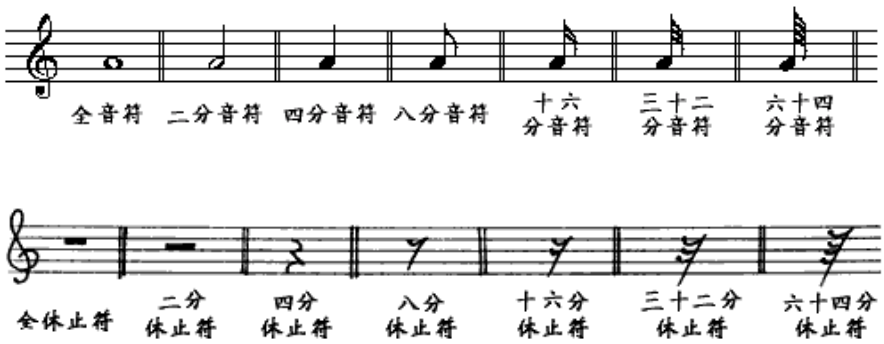

图 3.6　乐谱中的休止符和对应的乐音音符

在 MIDI 旋律中，休止符在音符曲线上表现为持续时间较长的音符，由于乐器在弹奏此类较长的音符时，乐弦只会触发一次，而随着时间的变化，振动的能量逐渐衰弱。当遇到休止符时停止弹奏，随时间的变化能量可能衰减到 0，并持续一定时间静音。人在哼唱过程中的停顿与休止符是一样的道理，如图 3.7 所示，展示了哼唱旋律中的停顿点与 MIDI 旋律中的休止符之间的对应关系，每一个停顿点对应了一个 MIDI 中的休止符。可见停顿点实际上是旋律中小节的分割点。

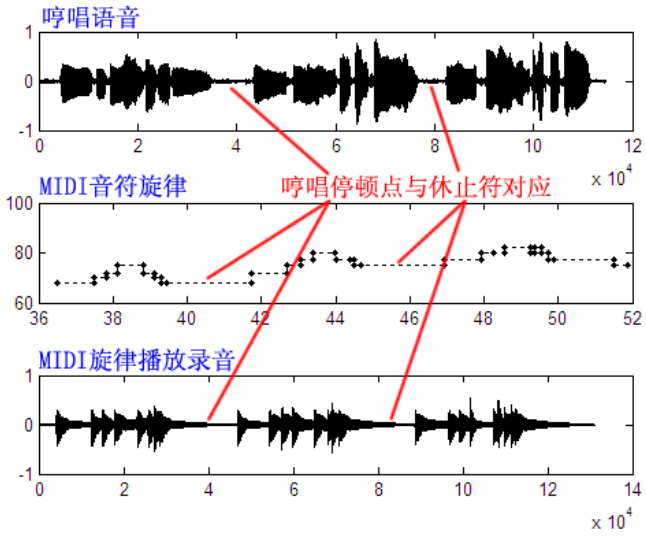

图 3.7　停顿点与 MIDI 中的休止符对应关系

音乐旋律中的停顿使得人在哼唱的过程中可以进行换气以保证哼唱的继续进行，同时也有时间来思考准备下一句的哼唱过程，而对于停顿点的时间，哼

唱者可能因为各种因素的影响，停顿的时间长度不再与目标 MIDI 旋律中的停顿时间保持线性变化的关系。同时不同哼唱的不同句子之间可能存在不同的线性变化参数。基于这两种假设，可以考虑在旋律匹配的过程中，以哼唱旋律的停顿点为分界点，并且在 MIDI 旋律中寻找最合适的对应分段方案，对哼唱旋律和 MIDI 旋律之间的对应片段进行线性伸缩匹配，将能够获得更好的匹配效果。

3.3.3　停顿点的估计

如图 3.5 中所示，停顿点在音高特征上的表现是在一定的时间段内不存在音高的信息，同时语音信号能量下降到最低的幅值。

对旋律中停顿点的估计，主要利用旋律中的静音片段，方法描述如下：

（1）按照指定的静音长度阈值，从哼唱旋律中找出静音时间长度大于该阈值的位置，作为候选停顿点；

（2）对上述提取的任意两停顿点，若两停顿点之间的时间间隔小于指定阈值，则对比两停顿点之间的静音长度，将静音长度小的一个去除；

（3）经上述处理后得到最终的停顿点列表。

由于一般哼唱的小节都会保持一定的长度，在哼唱旋律中相距较近的两个停顿点，其中一个可能是非停顿点，因此需要去除这些停顿点。在上述算法中，通过比较两个停顿点的静音长度，将长度较小的一个去除，这是因为静音长度大的比长度小的是真正的停顿点的可能性更大。

具体算法描述如下：

输入：哼唱旋律的音符序列$Q = q_1, q_2, \cdots, q_n$，最小静音长度$minSil$，停顿点最小间距$minDist$

输出：停顿点位置音符列表S

算法：

$for\ q_i\ in\ Q$

$\quad if\ isSil(q_i)\ and\ q_i.duration > minSil\ then$

$\quad\quad if\ len(S) == 0\ then$

$\quad\quad\quad S.append(q_i)$

$\quad\quad else\ if\ q_i.startTime - lastEle(S).end > minDist\ then$

$\quad\quad\quad S.append(q_i)$

$$
\begin{aligned}
&\quad else\ if\ q_i.duration < lastEle(S).duration\ then\\
&\qquad continue\\
&\quad else\\
&\qquad removeLastEle(S)\\
&\qquad S.append(q_i)\\
&\quad end\ if\\
&end\ if\\
&end\ for
\end{aligned}
$$

上述算法中的$isSil(q_i)$用于判断是否是静音音符，$lastEle(S)$取S中最后一次添加的音符，$removeLastEle(S)$表示从S中移除最后一次添加的音符。通过上述算法，可以通过一遍运算得到所有的停顿点音符。

对于 MIDI 旋律，由于 MIDI 文件中已经手动标记了每小节的起点，也就是每个句子的起始点，因此可以直接获取。如图 3.8 展示了哼唱语音和其旋律的停顿点标注以及目标 MIDI 旋律的停顿点标注。从图中可以看出，哼唱旋律与 MIDI 旋律在停顿点的音符时间比例与旋律的分段片段内的时间比例是明显不一致的。这也就说明通过停顿点来切分然后进行分段匹配是一种合理的方案。

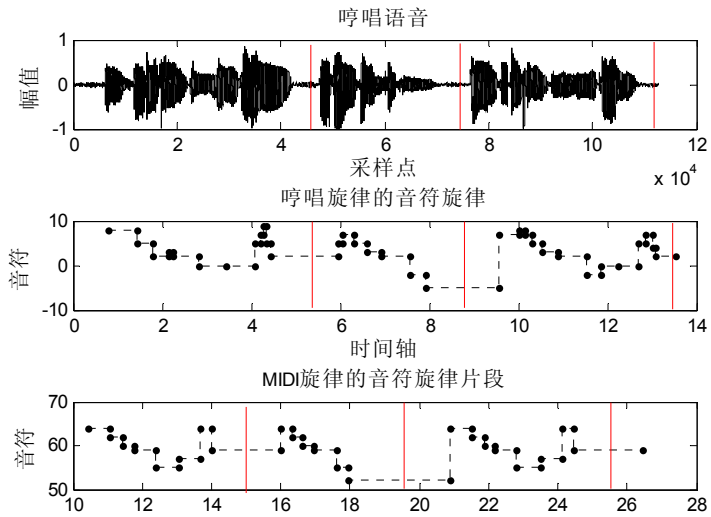

图 3.8　歌曲“冬季到台北来看雨”的前三句哼唱分段结果：根据停顿信息对哼唱及 MIDI 分段，哼唱旋律及目标 MIDI 旋律均被分成三个片段

3.3.4　估计参数的确定

在停顿点的估计算法中要用到两个参数，一个是最小静音间隔，另一个是两个停顿点的最小间隔。对停顿点的估计应当尽量的准确，如果最小静音间隔设置太小，则提取的停顿点过多，伪停顿点多，对分段匹配不利，而如果设置过大，则提取的停顿点太少，分段匹配的效果不明显。对于停顿点最小间隔也有类似结论。

如图 3.9 与图 3.10 所示，通过对 100 个哼唱语音进行标注停顿点，并进行统计来分析上述的静音间隔与停顿点间隔的分布。图 3.5 中，以 0.01 秒为间隔，统计了每个时间段上停顿点的个数，而在图 3.10 中，以 0.1 秒为间隔，统计了相邻停顿点之间时间间隔在对应时间范围内的样本个数。从统计结果可以看出，最小静音间隔的取值区间应当为$[0.1, 0.3]$较为合适，而相邻停顿点的最小间隔合适的取值范围是$[1, 2]$。当然由于理论上不能确定哪一个参数值对于分段匹配来说是最优的，具体参数值可以在实验中根据情况确定。

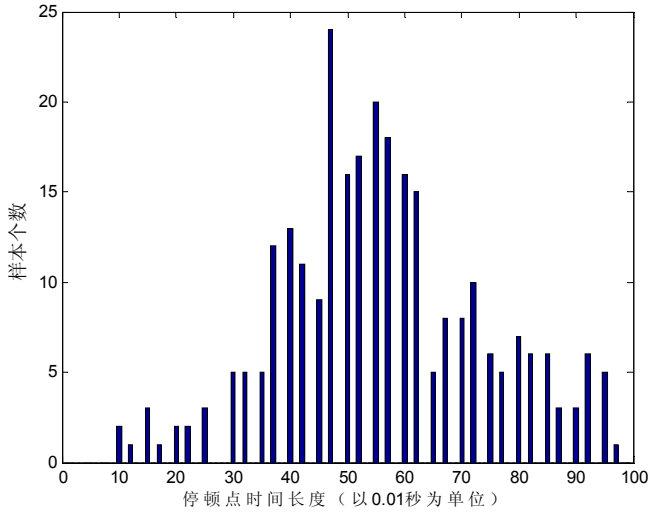

图 3.9　不同时间长度的停顿点标注样本数分布

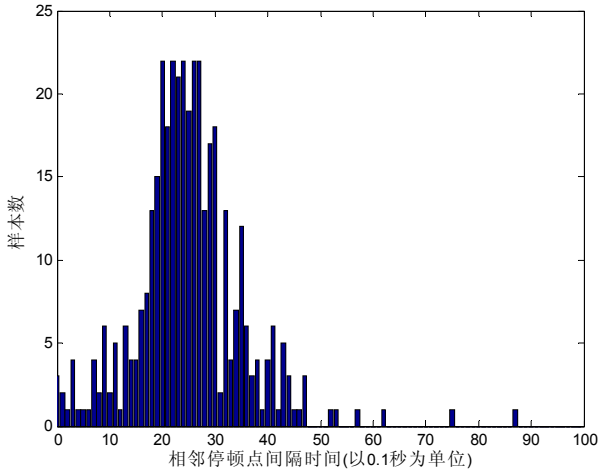

图 3.10　相邻停顿点间隔时间上标注样本的分布

3.4　实验

对启发式线性伸缩参数估计算法的描述见 0 节，本章中提出原有启发式估计算法由于起点单一可能无法得到全局最优参数，而极值点分段信息可用于增加启发式估计算法的候选起点来提高估计线性伸缩参数的准确性。为验证启发式估计算法可能无法得到全局最优线性伸缩参数的情况，本文设计了如下实验：

在进行线性伸缩参数估计时，使用两种方法。第一种是启发式估计算法，第二种是遍历方法，遍历方法即对拉伸系数及音高偏移的所有可能值都计算线性伸缩匹配分数并取匹配分数最优时对应的线性伸缩参数作为最终估计值。由于对哼唱旋律与目标 MIDI 旋律之间的线性伸缩参数估计越准确，则目标 MIDI 所在的排名越靠前，因此可以通过对所有候选进行参数估计后取最佳的线性伸缩匹配分数进行排名，以 Top-N 的准确率来衡量线性伸缩参数估计的准确性。

实验中所使用的数据库是 355 大小的哼唱数据库及 106 大小的 MIDI 数据库（数据库说明见 0 节）。

实验结果如图 3.11 所示，图中的横轴表示统计的 Top-N 中的不同 N 值的准确率，纵轴则表示对应的准确率。蓝色所标注的柱状图即为启发式估计的结果，红色标注的为遍历式估计的结果。图中遍历式估计算法得到的 Top-10 的准确率相比启发式估计算法由 0.85 提高到了 0.88，Top-30 对应的结果则从 0.92 提高到 0.94，由此可以看出，通过使用遍历所有可能参数进行线性伸缩参数估计的方

法能够使匹配结果中目标 MIDI 旋律所在的排名更靠前，对于与目标 MIDI 旋律之间的线性伸缩参数估计更为准确。这也就说明启发式估计算法估计得到的线性伸缩参数并不一定是全局最优的线性伸缩参数。

由以上分析表明，使用基于极值点的分段信息增加启发式线性伸缩参数估计算法的候选起点，以提高线性伸缩参数估计的准确性，是可行的。

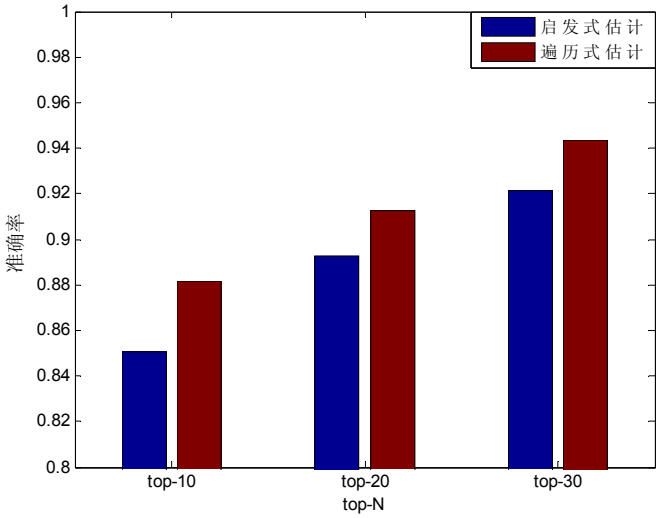

图 3.11　启发式估计算法与遍历式估计算法的准确率

对于基于停顿点的分段信息，可以用于对哼唱旋律进行分段的线性伸缩匹配，这种分段的线性伸缩匹配比 LS、RA 方法更有效的前提条件是，由停顿点分割的每一哼唱旋律片段与对应的目标 MIDI 旋律片段的线性伸缩参数是不一致的。由停顿点分割的片段即为哼唱的句子，为了验证哼唱旋律中的每一句子与目标 MIDI 旋律片段之间具有不同的线性伸缩参数，本文设计了如下的实验：

取一哼唱旋律以及对应的目标 MIDI 旋律，提取哼唱旋律的停顿点，对哼唱旋律中的每一停顿点，在目标 MIDI 旋律中找到对应的最佳分段点（手工），然后对哼唱旋律中的每一片段，计算与对应的目标 MIDI 旋律片段之间的最佳线性伸缩参数。这里的最佳线性伸缩参数的获取是通过遍历的方式得到的，拉伸系数的所有候选值是区间[0.5,2]中间隔为 0.05 的所有点，如 0.5,0.55,0.6 等。假设两旋律片段的第一个音符音高相差为b_0，则音高偏移的所有候选值是区间$[b_0-12, b_0+12]$中间隔为 1 的所有值。

最终得到的哼唱旋律上每一片段的参数估计结果如图 3.12 所示，图中的

ratio 表示两个片段估计得到的拉伸系数，shift 为估计得到的音高偏移，红色的箭头表示对应片段的起点，MIDI 旋律中的红竖线表示每一句的起始点位置。图中旋律曲线提取了 2 个停顿点，将哼唱旋律分成 3 段，每个片段与目标 MIDI 旋律片段之间的拉伸系数分别为 1.0, 0.9, 0.95，音高偏移则保持一致，均为 8。从这个结果可以看出，哼唱旋律中的不同片段与目标 MIDI 的旋律片段之间达到最佳匹配时的线性伸缩参数是不同的。因此可以说明，使用分段匹配的方法对不同的片段使用不同的线性伸缩参数进行匹配可以达到更好的匹配效果，说明了使用停顿点分段信息进行分段匹配的可行性。

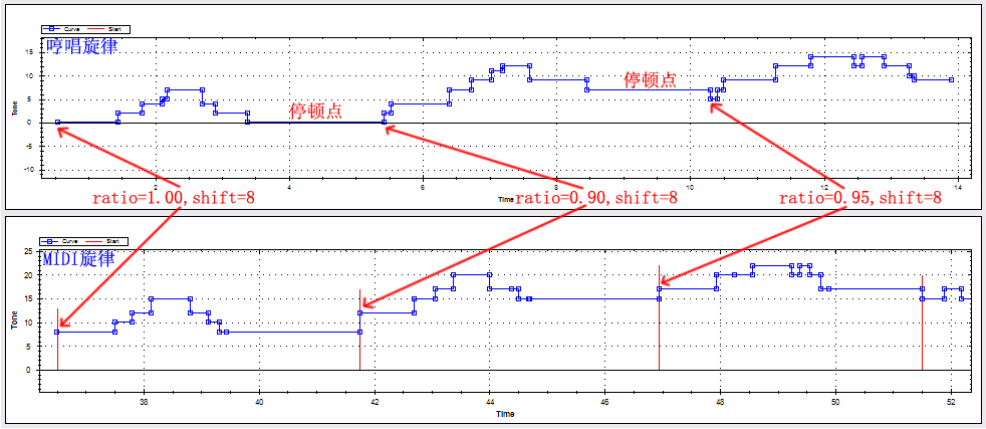

图 3.12　停顿点分割的每一片段的最优线性伸缩参数

由上述两个实验说明了极值点分段信息应用于启发式估计及停顿点分段信息应用于精确匹配中的分段匹配是合理的。

3.5　本章小结

本章介绍了基于极值点和停顿点的分段信息。对于基于极值点的分段信息，由于其对齐效果较好，可用于在启发式线性伸缩参数估计时增加候选起点，提高参数估计的准确性。而对于基于停顿点的分段信息，由于它反映了旋律的小节结构，而小节上哼唱旋律与目标 MIDI 旋律的线性伸缩参数保持稳定，小节切换时则不稳定，因此可以利用停顿点分段信息进行分段地线性伸缩匹配，达到更好的效果。下一章将介绍利用基于极值点的分段信息提高启发式估计参数的准确性及利用基于停顿点的分段信息进行分段地线性伸缩匹配。

第4章　基于分段信息的匹配算法

4.1　本章引论

在前文中介绍了基于极值点和基于停顿点的分段信息，基于这两种分段信息可以对旋律做不同的分段。基于极值点的分段信息，描述了旋律中低音向高音变化再变为低音的转换过程以及变化的程度和时间位置等。而基于停顿点的分段信息，其物理含义是休止符，通过分段信息描述了哼唱的旋律哼唱了多少个句子。

因此分段信息可以看作是旋律所表现出来的整体的、粗略的结构信息。对于这些信息我们可以直接用来进行简单匹配，或者对旋律进行适当的调整、对齐，或者对精确匹配起到框架的作用。简单匹配即单纯以分段信息计算哼唱旋律和 MIDI 旋律之间的差异，并去除一部分不可能的结果，又称为快速匹配，本文中我们不作考虑。而精确匹配中的框架作用，则表示在哼唱旋律和 MIDI 旋律之间进行匹配时，利用分段信息来控制两段旋律之间的整体对齐，对于局部的两个旋律片段之间，通过适当的调整进行线性伸缩匹配。

本章主要研究利用分段信息来对旋律进行更有效的匹配，包括归一化和精确匹配，同时也会通过在实验数据集上进行算法的测试，来比较通过加入分段信息的匹配过程和其他匹配算法之间的区别。本章的内容安排如下：第二节介绍基于极值点分段信息的启发式线性伸缩参数估计算法；第三节介绍利用分段信息的动态规划算法；第四节介绍利用分段信息的递归匹配算法；第五节给出实验的环境条件；第六节中对实验结果进行分析；最后在第七小节中对本章内容进行小结。

4.2　基于极值点分段信息的线性伸缩参数估计

在哼唱旋律和 MIDI 旋律匹配前，需要对哼唱旋律和 MIDI 旋律进行初步的归一化操作，以增加后面进行精确匹配时匹配到目标旋律的概率。归一化的过程是将 MIDI 旋律整体进行适当的线性拉伸及纵向平移，使得哼唱旋律与 MIDI 旋律之间的差异达到最小。如图 4.1，图 4.2 所示为哼唱旋律和 MIDI 旋律整体进行归一化前后的对齐结果，通过归一化，使得哼唱旋律与 MIDI 旋律有了基本的对齐结果，之后可以通过精确匹配的方法，对哼唱旋律和 MIDI 旋律进行更精确的对

齐并计算相似度。

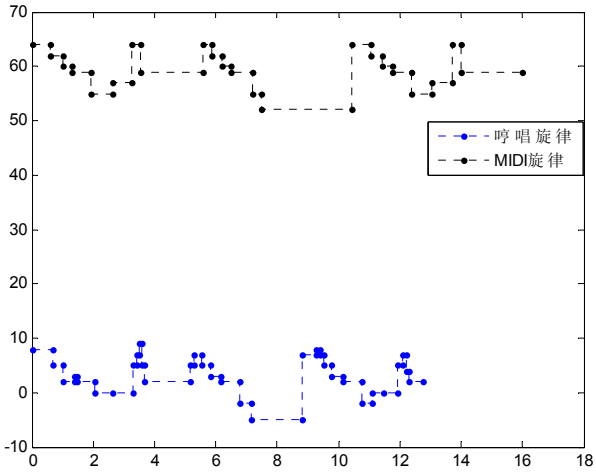

图 4.1　哼唱旋律与 MIDI 旋律未标准化前的对齐结果

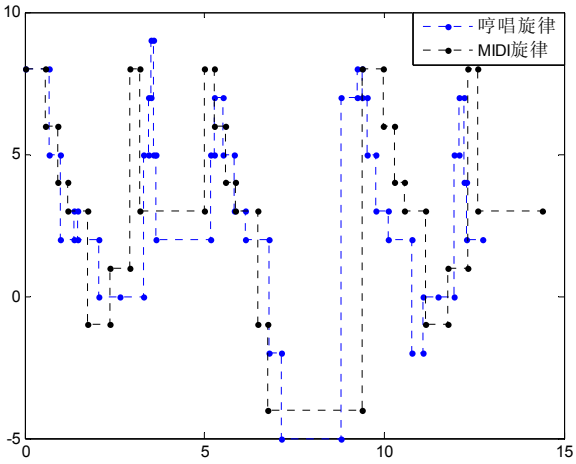

图 4.2　哼唱旋律与 MIDI 旋律经过标准化后的对齐结果

对旋律整体拉伸系数及音高偏移的估计本文采用了类似[3]中的算法，进行启发式的估计，所不同的是[3]中通过启发式方法进行多次的动态规划来计算精确匹配的分数，而本文则通过启发式进行多次的线性伸缩来估计最优的整体线性伸缩参数。具体的估计算法描述如下：

首先，初始化音高偏移$b = b_0$为哼唱旋律和 MIDI 旋律的第一个音符的音高差值，系统首先估计拉伸系数的最佳值，并且限定估计的区间为$[0.5, 2.0]$：

(1).初始化拉伸系数：$\alpha = \alpha_0 = 1.0$,并令

$$span = \begin{cases} 1.0/N, \ if \ \alpha > 1.0 \\ 0.5/N, \ if \ \alpha < 1.0 \end{cases}$$

这里N所代表的含义是将区间$[0.5, 2.0]$分割成$2 \times N$份，本文中设定为$N = 10$.
(2).分别使用拉伸系数α, $\alpha + span$,及$\alpha - span$三个参数计算哼唱旋律和数据库旋律间的距离（计算距离使用线性伸缩方法），假设计算的距离分别为d_0, d_1, d_{-1}。
(3).如果$d_0 = \min\{d_0, d_1, d_{-1}\}$或者$d_0$已经到达区间$[0.5, 2.0]$的边界值，停止算法；否则转(4)。
(4).更新α的值：

如果$d_1 = min\{d_0, d_1, d_{-1}\}$，则$\alpha = \alpha + span$;

如果$d_{-1} = min\{d_0, d_1, d_{-1}\}$，则$\alpha = \alpha - span$。

然后转第(2)步。

实际算法中，第(2)步计算得到的距离分数需要缓存起来，防止多次使用时造成重复计算。通过上述步骤计算得到了最佳的拉伸系数α。再经过类似的搜索过程来估计最佳的音高偏移，不同的是估计音高偏移时的取值序列为$[b_0 - M, b_0 + M]$，同时取值间隔为 1，这里的M表示从b_0往两侧搜索的半径。这样得到最优的拉伸系数和音高偏移后，就能对哼唱旋律和 MIDI 旋律进行归一化了。

上述的启发式估计过程，由于其启发式估计从一个单一的起点α_0, b_0开始估计，而线性伸缩的匹配分数与其拉伸系数及音高偏移参数之间并非单调的变化关系。因此在估计中可能找到局部的最优点，导致对参数的估计结果不够准确，这一点在 0 节中也已做证明。

可以通过增加启发式估计的候选起点来解决这一问题，从图 3.3 中可以看出哼唱旋律中的极值点与目标旋律中的极值点对应非常整齐，哼唱旋律和目标 MIDI 旋律中从起点到对应的极值点之间的旋律保持一致，因此可以利用这一特点来增加启发式估计时的候选线性伸缩参数值。

增加候选起点的基本思想是：

(1) 对哼唱旋律中的每一极值点E_i，在 MIDI 旋律中一定范围内搜索对应极值点。
(2) 如果对E_i找到对应极值点E_i'，则计算E_i与E_i'之间的音高差，并加入启发式估计的候选音高偏移参数列表，计算E_i相对哼唱旋律起点的时间与E_i'相对 MIDI 旋

律起点的时间之比，作为拉伸系数候选增加到启发式估计候选拉伸系数列表中。
(3) 对所有候选的拉伸系数，计算各自对应的线性伸缩匹配分数，并取分数最优者对应的拉伸系数作为启发式估计新的起点，对所有候选音高偏移系数作相同的处理。
(4) 使用新的拉伸系数和音高偏移进行启发式估计。

具体算法描述如下：

输入：

哼唱旋律的极值点列表$E_1, E_2, \cdots, E_n$

MIDI 旋律的极值点列表$E_1^{'}, E_2^{'}, \cdots, E_m^{'}$

哼唱旋律Q与 MIDI 旋律M

哼唱极值点搜索对应极值点时的搜索时间比例$[L_{min}, L_{max}]$，比如[0.5,2]。

输出：

极值点对的集合S

算法：

$for\ E_i\ in\ (E_1, E_2, \cdots, E_n)$
　$for\ E_j^{'}\ in\ (E_1^{'}, E_2^{'}, \cdots, E_m^{'})$
　　$if\ isMatch(E_i, E_j^{'}, [L_{min}, L_{max}])$
　　$S.append(E_i, E_j^{'})$
　　$end\ if$
　$end\ for$
$end\ for$

其中$isMatch(E_i, E_j^{'}, [L_{min}, L_{max}])$表示判断极值点$E_i$和$E_j^{'}$是否满足时间比例限制$[L_{min}, L_{max}]$。最终可以将集合$S$中每一极值点对的音高差和拉伸比例加入到启发式估计的候选起点中。

如图 4.3 是一个增加候选起点的示例，图中哼唱旋律有 3 个极值点，每个极值点都找到了对应的极值点，因此总共会对启发式估计增加 3 组候选参数。

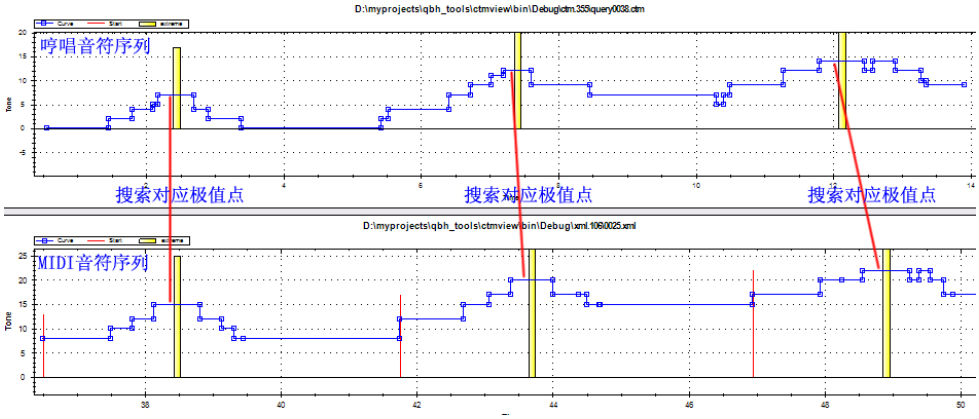

图 4.3　利用极值点分段信息增加启发式估计候选起点

4.3　基于停顿点分段信息的动态规划

基于分段信息的动态规划，是指根据分段信息对哼唱旋律进行分段，并按逐个片段进行匹配，对于每一个片段，根据其时间信息在 MIDI 旋律中找到对应的旋律片段，与 MIDI 旋律片段进行线性伸缩匹配，同时对 MIDI 旋律片段的分段点可以进行适当调整，构造对 MIDI 旋律的多种分段方案，通过动态规划，从这些分段方案中找到最佳的分段方案，同时获取最佳匹配分数。这种方法是在整体上进行动态规划，对片段而言则进行线性伸缩匹配。

这里的分段信息，即为前一章中所提出的基于停顿点的分段信息。通过这种分段匹配的方式，保证在每个分段上进行线性伸缩匹配，使用独立的拉伸系数和音高偏移，同时利用动态规划对 MIDI 旋律的分段边界点进行调整得到的多种方案，并在这些方案中选择最佳的分段方案。

如图 4.4 所示为一哼唱旋律与 MIDI 旋律匹配的情形，在哼唱旋律中检测到 2 个停顿点，将哼唱旋律分割成 3 段，图中的黄线表示停顿点的位置。在进行匹配时，对哼唱旋律从左往右，对第一个停顿点即图中的第一分段点，在 MIDI 旋律中找到对应的分段点，并且计算对应划分分段的线性伸缩匹配分数, 由此得到多种分段方案，接着对第二个分段点，也搜索 MIDI 中的候选分段点，计算匹配分数，假设对于图中的哼唱旋律的每个停顿点，在 MIDI 旋律中有 3 个对应的候选分段点，则对 MIDI 旋律可以有 9 种分段方法，动态规划的目的便是从这些分段方案中找出一种最好的分段方式。

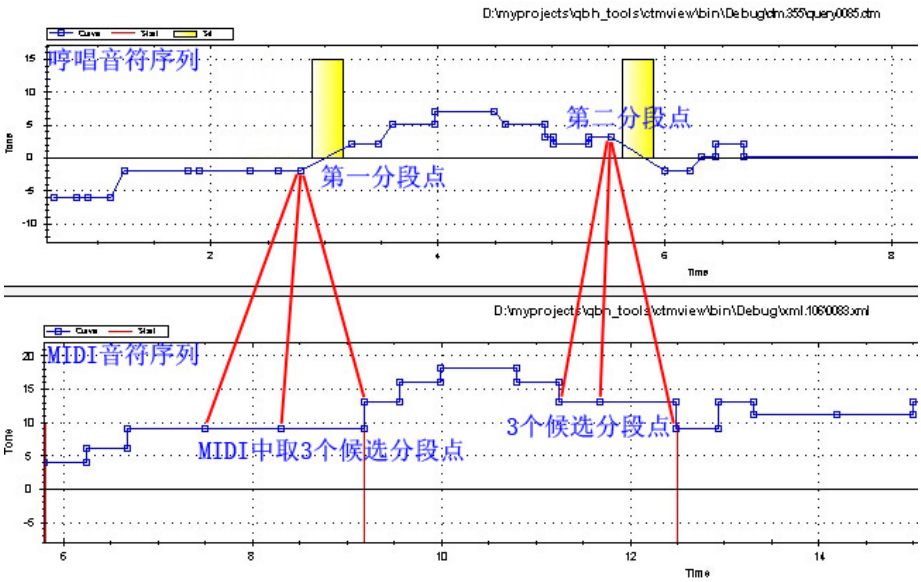

图 4.4　使用基于停顿点的分段信息进行动态规划匹配

假设哼唱旋律的音符序列为$Q=(x_1,x_2,\cdots,x_m)$，其中$x_i=(p_i^q,d_i^q)$表示哼唱旋律中的第i个音符。MIDI 旋律的音符序列为$M=(y_1,y_2,\cdots,y_n)$，其中$y_i=(p_i^m,d_i^m)$表示 MIDI 旋律中的第i个音符。同时假设两段旋律之间已经估计过拉伸系数和音高偏移并进行了归一化。则基于分段信息的动态规划算法可以描述如下：

$$DPLS(Q,M)=\min_{w\in cand(x_k)}(DPLS([x_1,\cdots,x_k],[y_1,\cdots,y_w])+ LS([x_{k+1},\cdots,x_m],[y_{w+1},\cdots,y_n])) \tag{4-1}$$

上述公式中的x_k表示哼唱旋律中最后一个停顿点对应的音符，而$cand(x_k)$则表示停顿点音符x_k在 MIDI 旋律中对应的候选分段点。上述公式的实际含义是：对哼唱旋律中的最后一个停顿点音符，从其在 MIDI 旋律中的对应候选音符中找出一个音符，按照这两个音符分别对两段旋律划分成两段，然后对左边进行动态规划，对右边进行线性伸缩匹配。最后从候选音符中找出一个使得两边匹配的分数和达到最小的候选音符作为最佳分段点。

在最终对旋律进行分段后进行线性伸缩匹配时，使用哼唱旋律的音高序列与 MIDI 旋律的音符序列进行匹配。

4.4　基于停顿点分段信息的递归匹配

基于停顿点分段信息的递归匹配方法和基于停顿点分段信息的动态规划方法类似，不同之处只是从另一个角度进行分段，二者的差别仅仅是按照不同的分段方式进行匹配。递归匹配每次将哼唱旋律和 MIDI 旋律分别分割成 2 段，并分别对两侧再进行递归匹配，直到达到指定的递归次数，或不可再分，此时直接返回线性伸缩匹配的结果。

递归匹配的方法相比动态规划的匹配方法在对旋律整体的把握上具有优势，它克服了动态规划的方法在匹配路径过长时容易导致丢失最佳路径的缺点。

递归匹配的方法最终也是在音高序列上计算匹配分数。设哼唱旋律的音高序列为$Q = (q_1, q_2, \cdots, q_n)$，其中$q_i$表示第$i$帧的音高，$n$表示哼唱旋律音高序列的总长度。设候选的 MIDI 旋律片段为$N = ((p_1, d_1), (p_2, d_2), \cdots, (p_m, d_m))$，其中$(p_i, d_i)$表示第$i$个音符的音高和时长，这里的时长以帧数表示，每帧窗宽与哼唱旋律的帧窗宽一致。设哼唱旋律中的停顿点对 MIDI 旋律按时间对齐后找到候选分段点后可继续选取的提取候选距离该分段点的音符个数范围为$[-L, L]$,设D为最大可递归的深度，一般设置为 3 或 4。则算法描述如下：

$function$ **RALS**$(\mathbf{Q}, \mathbf{N}, [-\mathbf{L}, \mathbf{L}], \mathbf{D})$:
$scorels = LS(Q, N)$
　$i \leftarrow mid(Q)$
　if $i < 0$ **then**
　　$return\ scorels$
　end if
　$j \leftarrow Near(i)$, $minDiff \leftarrow \infty$
　$Q_1 = (q_1, q_2, \cdots, q_i), Q_2 = (q_{i+1}, q_{i+2}, \cdots, q_n)$
　$N_1 \leftarrow ((p_1, d_1), \cdots, (p_j, d_j)), N_2 \leftarrow ((p_{j+1}, d_{j+1}), \cdots, (p_m, d_m))$
　$k = 0$
　for $l\ in\ [-L, L]$ **do**
　　$N_1' \leftarrow ((p_1, d_1), \cdots, (p_{j+l}, d_{j+l})), N_2' \leftarrow ((p_{j+l+1}, d_{j+l+1}), \cdots, (p_m, d_m))$
　　$score \leftarrow LS(Q_1, N_1) + LS(Q_2, N_2)$
　　if $score < minDiff$ **then**
　　　$minDiff \leftarrow score$
　　　$k \leftarrow j + l$

end if

end for

if $D = 0$ **then**

return $minDiff$

else

$N_1 \leftarrow ((p_1, d_1), \cdots, (p_{j+k}, d_{j+k})), N_2 \leftarrow ((p_{j+k+1}, d_{j+k+1}), \cdots, (p_m, d_m))$

return $\min(scorels,$

$RALS(Q_1, N_1, [-L, L], D-1) + RALS(Q_2, N_2, [-L, L], D-1))$

end if

上述算法中$mid(Q)$表示获取哼唱旋律中最接近中点的停顿点的位置，取停顿点的最后一帧位置，而$Near(i)$则表示在 MIDI 旋律中按时间对齐后找到与$mid(Q)$位置对应的位置，$LS(Q, N)$表示使用线性伸缩的方法对音高序列Q和音符序列N计算差异值，$LS(Q_1, N_1)$,$LS(Q_2, N_2)$亦同理。$RALS(Q_1, N_1, [-L, L], D-1)$则表示对子序列$Q_1$和$N_1$继续使用参数$[-L, L]$再做$D-1$层的递归调用。

在上述算法中，每一次对哼唱旋律取分段点时，取离中点最近的停顿点，如果递归过程中哼唱旋律已经没有停顿点可选，并且还没有达到递归的层次限制，则直接返回线性伸缩匹配的结果。

上述的递归过程中，不直接选择中点，而是选择离中点最近的停顿点。这是因为一般在哼唱时，在句子内部变化哼唱的速度和音高等特征的概率相对比较小，在句子内部哼唱旋律相对于目标 MIDI 旋律的拉伸系数和音高偏移而言比较稳定，而在两个句子之间的换气点，则可能是两个句子之间线性伸缩参数变化的分界点，比如一个人哼唱了一个句子后，经过换气，哼唱的音调和速度就可能出现稍微的变动，这取决于每个人的哼唱水平。总之对于哼唱而言，在句子之间变换速度和音高的可能性比在句子中间变化的可能性要大。因此，这里在递归匹配中，我们使用停顿点作为分段的边界，而不是直接使用中点。由算法可以看出，当递归到一定程度，可能旋律过短不再有任何停顿点分段信息，算法将直接进行线性伸缩匹配并返回对应分数。因此，算法在最坏的情况下，没有任何分段点可分时，与线性伸缩的方法性能相同。

如图 4.5 所示为一段哼唱旋律与 MIDI 旋律进行匹配的例子。图中的哼唱旋律具有三个停顿点，将旋律分成 4 段，其中黄线标注的是停顿点的位置。在递归匹配开始时，先对哼唱旋律和 MIDI 旋律做一遍线性伸缩匹配，再取哼唱旋律的中

间停顿点，即图中的第一分段点，对哼唱旋律进行分段，对应的在 MIDI 旋律中找到多个候选分段点，将 MIDI 旋律分成对应的两段。然后对哼唱旋律和 MIDI 旋律进行分段匹配，并保留最优的分段方案及结果，与直接线性伸缩的结果相比较，返回较好的结果。在分段匹配过程中，需要计算两侧的递归匹配分数，同样也是按中间停顿点进行分段的方法，如图中的第二分段点和第三分段点，从而计算得到左右两侧的递归匹配分数。

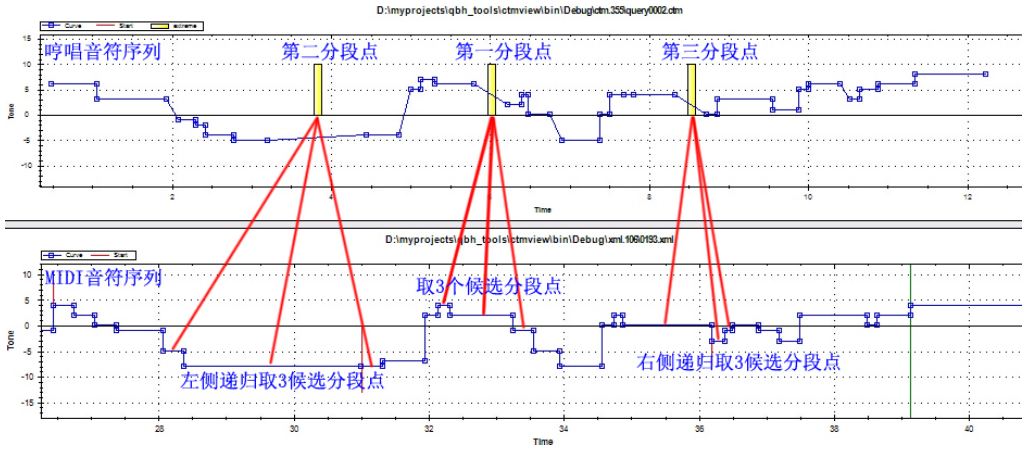

图 4.5　基于停顿点分段信息的递归匹配示例

4.5　系统结构与实验数据

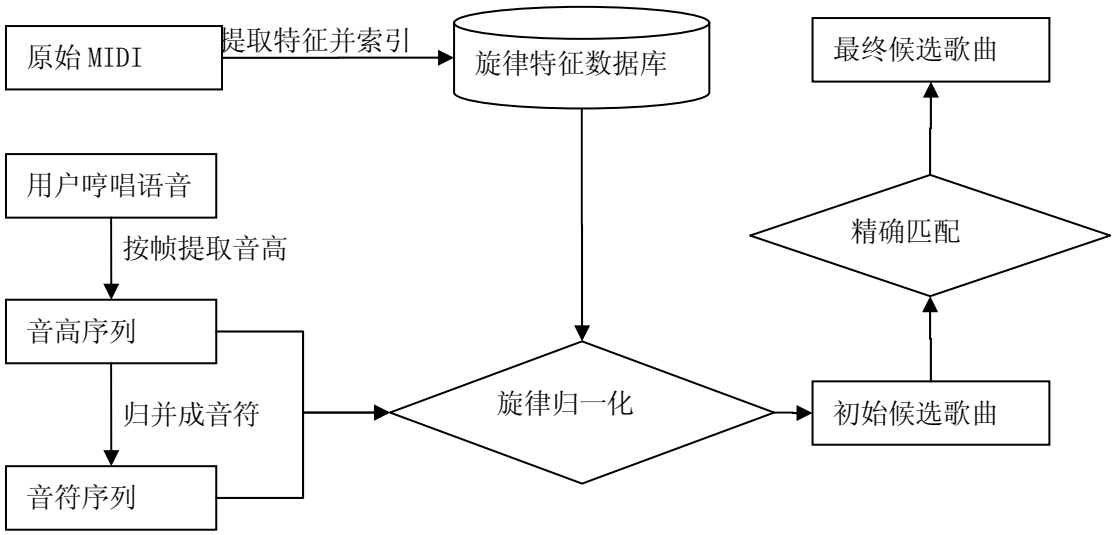

图 4.6　QBH 系统的结构及匹配流程图

本文使用的 QBH 系统的流程图如图 4.6 所示，系统主要由四个模块组成：

（1）MIDI 旋律数据库模块

MIDI 旋律数据库模块的主要功能是提取 MIDI 文件的旋律信息并提取特征，对特征进行索引。本文所使用的 MIDI 信息为 XML 格式的 MIDI，这种 MIDI 可以是通过一款称为美得理简朴制作软件的工具录制而成，由软件录制的 XML 格式的 MIDI 信息再经过简化处理便得到最终的表示 MIDI 的 XML 文件。XML 文件的格式如下：

```
<song>
<melody>
<name>0065.mid_0</name>
<note>
<track>1</track>
<eventindex>1</eventindex>
<tone>57</tone>
<start>26.689583</start>
<end>27.129166</end>
<duration>0.439583</duration>
<possibility>1.0</possibility>
</note>
<note>
<track>1</track>
<eventindex>3</eventindex>
<tone>59</tone>
<start>27.129166</start>
<end>27.498958</end>
<duration>0.369792</duration>
<possibility>0.0</possibility>
</note>
…
</melody>
</song>
```

XML 格式中的<note>标签表示一个音符，<tone>标签表示 MIDI 中音符的音高,<start>表示音符的起始时间，<end>表示音符的结束时间, <duration>表示持续时间, <possibility>表示是否为句子的起始点，当该值为 1.0 时表示句子的起始点，

否则为非起始点。

经过处理后的 XML 文件经过提取音符、归并音符、提取极值点并做索引等操作，便构成了 MIDI 数据库。

在匹配过程中，MIDI 数据库的作用是生成用于匹配的候选 MIDI 旋律，由于数据库中的 MIDI 标注了句子的起始点，而且本文中我们假设哼唱查询语音都是从歌曲的某一句子开头开始哼唱的，因此数据库生成候选时，按照每首 MIDI 的句子起始点标注信息，对每一个句子起始点标注生成一个候选旋律，用于匹配。这样生成的候选旋律只指定了旋律的起点，而终点没有指定。

（2）哼唱输入处理模块

哼唱输入处理模块的主要功能是对哼唱输入的语音进行分帧，并估计每一帧的基频，经过转化后得到音高的序列，这个序列再经过一定的算法处理，归并成音符序列，由此得到两种序列用于匹配算法。在本文中音高特征及音符特征的提取使用的是文献[36]中的算法，这种方法在 0 节中已做相关介绍，音高序列的提取使用一种改进的自相关算法提取得到，在音高提取之前，哼唱的语音通过基于统计的音符转化算法被解码成音符和静音的片段序列，和传统的自相关方法在全局检索基频峰值不同的是，这里的基频提取的算法只在解码得到的音符的局部区域内检索相关峰值，这种方法对于带噪声的输入语音具有更好的鲁棒性。

（3）旋律归一化模块

在进行旋律的精确匹配前，由于哼唱旋律和数据库旋律间存在整体的速度及音高的不一致，需要对哼唱旋律与数据库旋律进行音高偏移和拉伸系数估计，以将数据库旋律规整为与哼唱旋律一致的速度及音高。在本文的系统中，这一过程是通过一个启发式的搜索过程完成的，具体的算法请参考 0。

尽管这一模块的主要功能是对数据库旋律进行规整，但在这一过程中已经对旋律整体做了线性伸缩匹配，因此可以使用此步中得到的最优匹配分数对初始候选集合进行排序，然后排除一部分不可能的结果。因此这一模块同时也作为候选过滤模块使用。

（4）精确匹配模块

精确匹配模块的主要功能是对旋律规整后剩余的候选旋律进行更精确的匹配

分数计算，并按照最终得到的匹配分数排序，给出 Top-20 候选作为系统的输出结果。精确匹配是本文的研究重点，具体算法已在 0 节，0 节中介绍。

本文使用的哼唱测试数据库总共由 355 个哼唱语音组成，哼唱语音的格式为 wav 文件，哼唱语音以 8kHZ 的采样率 16 比特编码录制得到，总长度约为 71 分钟，平均每个哼唱语音的长度为 12.4 秒。数据库中的哼唱语音多数是以带歌词的形式哼唱的，哼唱者都是一般音乐水平的人，没有经过特殊的音乐知识训练。这个数据库由中科院声学所中科信利实验室提供，曾经被作为 2008 年的 MIREX QBH evaluation[45]中的一个标准测试集。需要说明的是，所有的哼唱语音均是从歌曲中某一句子的开始处开始哼唱，一般大致哼唱 3~4 个句子，但是哼唱语音可能会在某一句子的中间结束。

本文实验中所使用的 MIDI 数据库有 2 个，一个总共包含 5223 首歌曲，该 5223 的数据库由我们自己使用录制工具录制得到。数据库中包含 106 首为哼唱数据库的目标 MIDI 歌曲，MIDI 数据库中所有的 MIDI 旋律都标注了 MIDI 旋律中的句子起始点，其中 106 首目标歌曲被标注成了 2213 段。另一个数据库则是 106 首目标 MIDI 数据库，该数据库属于 5223 数据库的子集，主要用于测试算法的参数。

4.6　实验验证及结果分析

4.6.1　评价准则及各算法描述

本文中提出利用极值点分段信息增加启发式估计线性伸缩参数的候选起点，同时提出利用停顿点分段信息进行动态规划以及递归匹配的方法，对此使用实验来验证算法的合理性。

在实验中，匹配的准确率可以使用 Top-1 及 MRR 来衡量，MRR 是在 MIREX 的 QBH 评测中对系统准确率的一种标准的评价方式，可以通过目标 MIDI 所在的排名来计算，计算公式如下：

$$MRR = (\sum_{q=1}^{N} c_q)/N \qquad (4\text{-}2)$$

这里c_q表示第q个哼唱查询得到的返回结果中其目标 MIDI 所在排名的倒数，但如果目标 MIDI 不在前 20 的返回结果中，则对应的c_q的值为 0。可以看出，MRR的取值范围在$[0, 1]$内。

本文也实现了前面介绍的线性伸缩(LS)、动态规划(DP)、递归的线性伸缩(RA)

的方法作为对比，表 4.1 给出了各算法的简要介绍。LS 的方法直接进行整体的线性伸缩匹配，DP 算法作为传统算法，研究得比较多，本文也实现作为参考，本文实现的动态规划算法是带约束的动态规划算法，即对满足要求的路径上的每一个点，对应的哼唱旋律和 MIDI 旋律之间的时间比必须在指定的范围内，一般限定的范围为[0.5,2]。RA 方法与本文中 RALS 的方法比较类似，区别是 RA 的方法直接以其中一旋律的中点进行分段匹配，而 RALS 的方法则是以最靠近中点的停顿点作为切分点进行分段匹配。

表 4.1　各算法的介绍

算法简写	算法介绍
LS	线性伸缩算法，见 0 节中描述。
DP	动态规划的方法，具体算法见 0 节。
RA	递归匹配的方法，以中点切分递归匹配，见 0 节。
DPLS	本文方法，基于停顿点分段信息的分段地线性伸缩匹配，同时用动态规划计算最佳分段方式。
RALS	本文方法，基于停顿点分段信息的分段线性伸缩匹配，同时使用递归的方式获取最佳的分段方式。
DPLS_EP	本文方法，在 DPLS 的基础上，基于极值点分段信息增加启发式估计线性伸缩参数的候选起点对旋律进行归一化。
RALS_EP	本文方法，在 RALS 的基础上，基于极值点分段信息增加启发式估计线性伸缩参数的候选起点对旋律进行归一化。

4.6.2　基于停顿点分段信息的精确匹配

为了验证基于停顿点分段信息进行分段匹配的有效性，对 0 中列出的各算法，本文在前述 355 大小的哼唱数据库和 5223 大小的 MIDI 数据库上进行实验，系统的其他模块保持一致，精确匹配算法分别使用 LS、DP、RA、DPLS 以及 RALS 共 6 种算法。测试得到各算法的准确率如表 4.2 所示，图 4.7 给出了更直观的图示。从图 4.7 的结果可以看出，LS、DP、RA、DPLS 及 RALS 对应的准确率逐渐升高，Top-1 的准确率基于停顿点分段信息的匹配算法最好性能相比 LS、DP 及 RA 算法分别提高了 17%、14.7%及 4.8%，而对应的 MRR 提高分别是 15.3%、12.9%、4.8%。

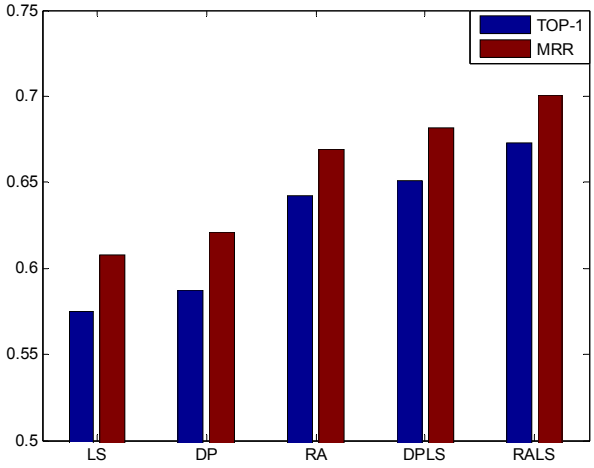

图 4.7　LS、DP、RA、DPLS、RALS 算法的性能比较

表 4.2　各算法的实验结果

	LS	**DP**	**RA**	**DPLS**	**RALS**
Top-1	0.575	0.587	0.642	0.651	0.673
MRR	0.608	0.621	0.669	0.682	0.701

由实验结果可知，DPLS 和 RALS 的方法优于 LS、DP 及 RA 三种方法。DP 优于 LS 的方法，是因为 LS 的方法对哼唱旋律以整体的方式与 MIDI 旋律进行伸缩匹配，导致其估计得到的拉伸系数及音高偏移过于粗略。而 DP 方法由于在实现时使用了一定的方法控制哼唱旋律与 MIDI 旋律之间匹配的速度比在一定的范围之内，因此比 LS 的方法更好。而 RA 的方法较 LS 的方法不同的是，它使用递归的方式分段的估计并优化线性伸缩匹配，更好地把握了旋律的整体轮廓进行匹配。DPLS 及 RALS 均优于前面的三种算法，DPLS 及 RALS 都限定了线性伸缩匹配的过程以句子为单位进行，并且对句子的边界作动态调整估计最佳的句子分段方式，两种算法优于 LS、DP 算法说明，基于分段信息进行分段地线性伸缩匹配算法更有效。而 DPLS、RALS 优于 RA 的方法也说明，基于停顿点进行分段比基于中点进行分段的方法更合理。另外，RALS 优于 DPLS 表明递归匹配这种自顶向下的分段方式优于动态规划从前往后的分段方式。

本实验中的实验结果证明了停顿点分段信息在分段匹配中的有效性，也印证了本文在第 3 章中的分析。

4.6.3　候选分段点数对动态规划分段匹配的影响

在 DPLS 算法中，对哼唱旋律中的每一个停顿点，需要在 MIDI 旋律中取多个候选分段点对 MIDI 旋律进行对应的分段。候选的分段点数越多，算法消耗的时间就越长，在实验中每一次对哼唱旋律中的一个停顿点搜索了 MIDI 旋律的分段后，会对所有的分段方案得到的匹配分数进行排序，并保留前 N 个最好的分段结果，然后继续进行下一个停顿点搜索。

本文实验中考虑了候选分段点数对动态规划的影响，通过在前述 355 大小的哼唱数据库及 5223 大小的 MIDI 数据库上进行实验，得到的结果如表 4.3 所示，表中的 beam 表示对每一个停顿点计算所有 MIDI 可能的分段方案后保留前 beam 个分段方案。t 则表示候选分段点的搜索范围，比如 t=1 则表示通过时间对齐在 MIDI 中找到与哼唱旋律中的停顿点对应的分段点后，将其左侧及右侧的 1 个音符作为候选分段点，因此总共的候选分段点数是 3 个。TIME 表示算法的平均响应时间。由表中的结果可以看出，DPLS 算法的准确率并非随候选分段点数的增加而提高，算法在 t=2 时的准确率优于 t=1 及 t=4 时的准确率，而计算时间则随着 t 的增大而增大。算法准确率不随 t 增大而提高的原因是获取候选分段点时并未考虑分段的时间比率是否满足要求，在本文中，进行精确匹配前已经对 MIDI 旋律进行了归一化处理，进行了适当的拉伸。当 t 过大时，会导致某些候选分段点切分的片段与哼唱旋律的对应片段之间的拉伸比例超出了合理的范围（本文认为[0.5,2]为合理的拉伸比例），反而导致引入错误的匹配结果。实验结果中 t 参数的变化导致的时间消耗的相对增加比例不大，这是因为在本文的系统中，主要的时间消耗集中在归一化旋律的处理过程。

由以上分析可知，在分段匹配过程中获取候选分段点时应当使得对应片段的拉伸比例满足在合理的范围内。

表 4.3　候选分段点数对 DPLS 的影响

	beam=5,t=1	**beam=5,t=2**	**beam=5,t=4**
Top-1	0.5803	0.6225	0.6197
MRR	0.6173	0.6533	0.6532
TIME	27.45	26.61	28.05

4.6.4　递归分段匹配准确率与递归层次的关系

递归匹配的方法通过递归的方式逐步对旋律进行分段进行匹配而达到最优化的效果，随着递归层次的增大，计算的复杂度也相应的增大。为研究递归匹配算法的准确率与递归的层次之间的关系，本文在上述的 355 大小的哼唱数据库以及 106 大小的 MIDI 数据库上进行实验，计算不同的递归层次下的算法性能，系统中其他模块的配置保持一致。实验中我们仅测试了 RA 算法，RALS 的方法与 RA 的分段方法基本一致，区别之处是 RALS 使用停顿点进行分段而 RA 使用中点进行分段。

实验结果如图 4.8 和图 4.9 所示。在图 4.8 中，本文使用了目标覆盖率这一指标，即 Top-N 准确率。图中的横轴表示返回的前 N 个候选，纵轴表示对应的目标覆盖率。从图中横轴上 0 到 50 之间的结果可以看出，递归层次在 1、2、3 时对应的结果差异比较大，而当递归层次超过 3 时，对应的目标覆盖率基本相差不大。图 4.9 则给出了递归层次为 1-5 时的 Top-1 与 MRR 随递归层次的变化情况，随着递归层次的增加，Top-1 准确率不断上升，但上升的幅度逐渐趋于平缓，特别是从递归 4 层增加到递归 5 层时，准确率的提高已不明显。

上述的结果是由哼唱数据库的特点决定的，在实验使用的 355 大小的哼唱数据库中，几乎所有的哼唱语音哼唱的句子数都不超过 5 句，大部分是哼唱 3 句或 4 句。因此当使用递归匹配的方法进行分段，分段超过一定次数时，再增加分段数则相当于对句子的片段再进行分段，而由于句子内部的线性伸缩参数是保持稳定的，因此分段匹配的效果不再明显，也导致准确率提高幅度变小。

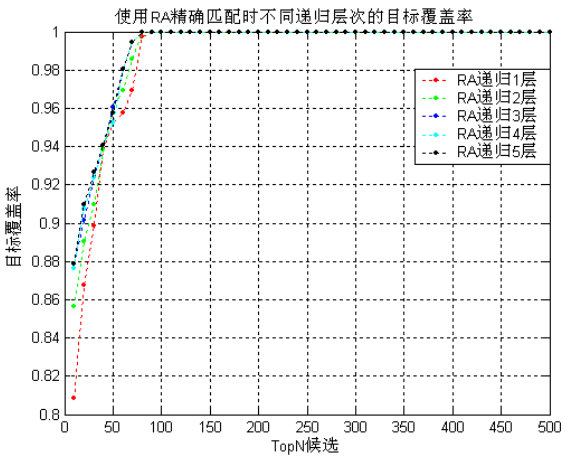

图 4.8　递归匹配方法的目标覆盖率与递归层次的关系

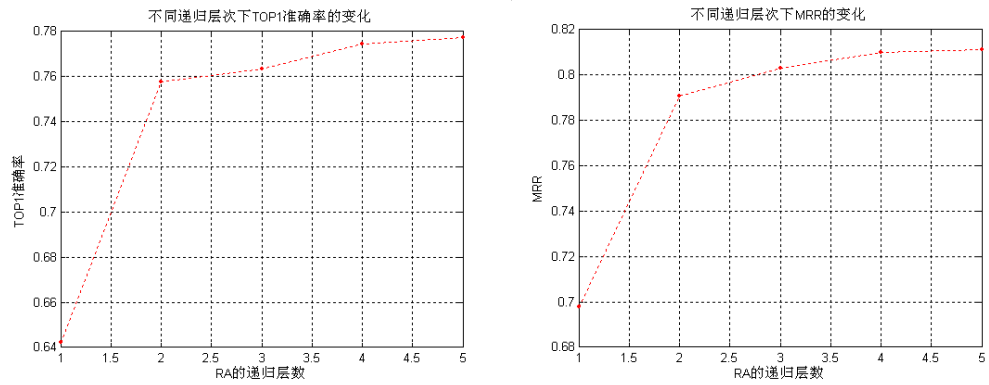

图 4.9　递归匹配方法的 Top-1 及 MRR 准确率与递归层次之间的关系

4.6.5　基于极值点的启发式线性伸缩参数估计算法

为了证明基于极值点分段信息增加启发式估计算法中的候选起点的有效性，本文设计了一组对比实验。对 DPLS 及 RALS 两种算法在精确匹配前的归一化操作，分别使用原有的启发式估计算法及新的启发式算法进行实验。实验数据集使用前述的 355 大小哼唱数据库及 5223 大小的 MIDI 数据库。

实验结果如表 4.4 所示，DPLS 及 RALS 表示不使用基于极值点的分段信息时的系统性能，DPSL_EP 及 RALS_EP 则表示对应的在启发式线性伸缩参数估计算法中使用基于极值点的分段信息增加候选起点后的系统性能。使用 RALS 作为精确匹配算法时新的启发式估计算法使系统的 Top-1 和 MRR 准确率分别提高了 1.8%和 3.3%。这是因为当使用基于极值点的分段信息增加了启发式估计候选起点后，有效地避免了在启发式估计线性伸缩参数的过程中陷入局部极值点。在第 3 章的实验中已经证明了启发式估计算法陷入局部最优解是可能的，因此本实验中的结果也是合理的。

通过本实验证明了基于极值点的分段信息在估计线性伸缩参数中的有效性。

表 4.4　新的启发式估计算法与原有估计算法对应系统性能对比

	DPLS	**RALS**	**DPLS_EP**	**RALS_EP**
Top-1	0.651	0.673	0.670	0.685
MRR	0.682	0.701	0.693	0.724

4.6.6　实验小结

在 0 节的实验中，基于停顿点分段信息的分段匹配算法相比传统的 LS 算法在 Top-1 上最高提高了 17%，而在 0 节中，使用新的启发式估计算法后，还可以将 RALS 的 Top-1 提高 1.8%。两个实验验证了本文提出方法的有效性。同时同为分段匹配方法的 RALS 和 RA 方法的对比实验表明，在同等条件下，RALS 相比 RA 在 Top-1 上从 0.642 提高到 0.673，相对提高了 4.8%，证明了以停顿点进行分段匹配的方式比以旋律中点进行分段匹配的方式更有效。因此，上述的实验充分证明了本文提出方法的合理性和有效性。

4.7　本章小结

在本章中，为了验证基于极值点分段信息对于启发式线性伸缩参数估计的有效性以及基于停顿点的分段信息对于分段匹配的有效性。本文设计了三种算法，基于极值点的分段信息增加启发式估计算法中的候选起点，同时利用基于停顿点的分段信息对旋律进行分段匹配，而分段匹配的过程使用了动态规划和递归匹配的方式来对与哼唱旋律对应的 MIDI 旋律的分段方式进行优化。通过在 355 大小的哼唱数据库以及 5223 大小的 MIDI 数据库上进行实验。实验结果表明，无论是利用基于极值点进行启发式线性伸缩参数估计的算法，还是基于停顿点分段信息进行动态规划或递归分段匹配的算法，均有利于旋律之间的匹配准确率的提高。从而说明，基于极值点的分段信息与基于停顿点的分段信息是两种有效的分段信息，同时基于这两种分段信息的匹配算法也是更有效的匹配算法。

第5章　结论与展望

哼唱检索相比传统基于描述信息的音乐检索方式更加人性化。线性伸缩方法是用于计算哼唱检索中旋律匹配分数的一种较好的算法，针对线性伸缩方法的启发式估计算法起点单一的问题及旋律局部线性伸缩参数不精确的问题，本文提出了两种分段信息，一种是基于极值点的分段信息，另一种是基于停顿点的分段信息。由于极值点位置在哼唱旋律和目标 MIDI 旋律之间的对应性，有利于估计线性伸缩的参数。停顿点分段信息则反映了哼唱旋律中的小节（句子）信息，停顿点对应于乐谱中的休止符，按小节的方式进行分段匹配更为合理。

本文研究了利用极值点分段信息来提高对哼唱旋律与 MIDI 旋律之间线性伸缩参数估计的准确性，通过利用极值点分段信息增加启发式估计算法中的候选起点来防止估计算法陷入局部最优解。本文同时也研究了对旋律基于停顿点分段信息进行分段匹配，由于旋律中的速度音高等描述旋律的重要参数在句子内部保持稳定，而在句子之间切换时易发生改变，因此以句子为单位进行的分段匹配更适用于基于线性伸缩的匹配方式。更进一步的，本文还探讨了使用动态规划和递归匹配两种不同的方式来获取对 MIDI 旋律的最佳分段方式，通过搜索对 MIDI 旋律的最佳分段方式，将哼唱旋律和 MIDI 旋律进行一致的分段，进行分段匹配，达到最优的匹配结果。

通过在包含 355 个哼唱语音的哼唱数据库及包含 5223 首 MIDI 的旋律数据库上进行实验，实验结果表明，使用 LS、DP、RA 方法得到的 Top-1 准确率分别为 0.575、0.587 及 0.642,对应的 MRR 分别为 0.608、0.621 及 0.669,而在最好的情况下，基于停顿点分段信息的分段匹配算法比 LS、DP、RA 算法在 Top-1 上分别提高 17%、14.7%和 4.8%，相应的 MRR 提高分别为 15.3%、12.9%和 4.8%。另外，使用 RALS 作为精确匹配算法时的 Top-1 和 MRR 准确率分别为 0.673 和 0.701，使用新的启发式估计算法后系统的 Top-1 和 MRR 准确率分别提高了 1.8%和 3.3%。RALS 和 RA 方法的比较也说明以停顿点作为分段点进行分段匹配的方式比直接以中点进行分段的分段匹配方法效果更好。另外，在基于停顿点的分段匹配方式中，使用递归的方式进行分段匹配比使用动态规划的效果好，说明了递归的方式对整体匹配的把握比动态规划的方法要好。

本文提出的 RALS 的方法相比传统的 LS 方法在 Top-1 上提高了 17% 。相比

传统方法，本文方法的优点是既利用了启发式估计快速的特点，又通过增加候选起点保证了线性伸缩参数估计的准确性。本文方法的另一优点是通过利用停顿点对旋律进行合理地分段匹配，既解决了传统方法中局部线性伸缩参数不准确的问题，又使得对局部参数的估计达到最优。

实验结果验证了极值点分段信息在线性伸缩参数估计中的有效性以及停顿点分段信息在分段匹配中的有效性，同时也说明了基于停顿点分段信息的分段匹配方法是更有效的旋律匹配算法。

但本文的研究工作仍然有很多需要改进的地方，主要包括：

（1）极值点分段信息的提取

对于基于极值点的分段信息，其提取的结果会受到旋律不稳定因素的影响，由于旋律的不稳定因素可能导致提取的特征中包含伪最高点或者去除了不该去除的极值点，因此在本文中仅用来增加启发式估计线性伸缩参数的候选点。但极值点的信息反映了旋律整体的变化趋势，是描述旋律结构很重要的信息，如果同时利用停顿点和极值点的信息来对旋律进行对齐，应该可以达到更好的效果。

（2）停顿点的估计

本文中的基于停顿点的分段信息使用了按最小静音间隔提取并进行归并的方法。最小静音间隔是绝对值，对不同的人，其哼唱的方式有所不同，自然也会导致这种按时间间隔提取的方式受到不同的影响。对于哼唱音色较好，哼唱的句子比较连续，同时句子之间的换气间隔比较明显的情况下，这种提取方式能够获得准确的分段信息。但假如哼唱者的声音沙哑或者哼唱时因无法发颤音等情况导致句子内部也出现静音时，这种方式提取的停顿点就多于实际的停顿点数量，从而会导致对匹配过程造成不利的影响。因此，如何更好地提取哼唱过程中的停顿信息，还需要进一步的研究。

（3）停顿点信息的利用

停顿点的信息在旋律中实际上代表了旋律小节的结束，是一类重要的参考点。在旋律的匹配过程中，也可以利用此类特征点做其他的匹配。比如利用停顿点附近的旋律特征对候选进行快速过滤，或者直接根据哼唱旋律中的停顿点

位置信息对候选进行快速过滤。再者，也可以考虑对停顿点分割的片段进行特征提取，并进行索引，通过索引特征达到快速匹配的目的，这种方式类似于乐纹的方法。

参考文献

[1] Asif Ghias, Jonathan Logan, David Chamberlin, et al. Query By Humming -- Musical Information Retrieval in an Audio Database in: ACM Multimedia 95 - Electronic Proceedings. 1995.

[2] Rodger J. McNab, Lloyd A. Smith, Ian H. Witten, et al. Tune Retrieval in the Multimedia Library. Multimedia Tools and Applications, 2000, 10(2-3).

[3] Jyh-Shing Roger Jang, M.-Y. Gao. A Query-by-Singing System based on Dynamic Programming. in: International Workshop on Intelligent Systems Resolutions (the 8th Bellman Continuum). 2000. 85~89.

[4] Jyh-Shing Roger Jang, Hong-Ru Lee, M.-Y. Kao. Content-based Music Retrieval Using Linear Scaling and Branch-and-Bound Tree search. in: Proc. of IEEE International Conference on Multimedia and Expo. 2001.

[5] Jyh-Shing Roger Jang, H.-R. Lee. Super MBox: An Efficient/Effective Content-based Music Retrieval System1. in: Proceedings of the Ninth ACM International Conference on Multimedia. 2001. 401~410.

[6] 李扬, 吴亚栋, 刘宝龙. 一种新的近似旋律匹配方法及其在哼唱检索中的应用. 计算机研究与发展, 2003, 40(11).

[7] Jonah Shifrin, W. Birmingham. Effectiveness of HMM-based Retrieval on Large Databases. in: Proc. of International Symposium of Music Information Retrieval (ISMIR). 2003.

[8] Jonah Shifrin, Bryan Pardo, Colin Meek, et al. HMM-Based Musical Query Retrieval. in: Joint Conference on Digital Libraries. 2002.

[9] 陈知困, 徐明, 黄云森. 一种高效的基于 CHMM 的哼唱式旋律检索方法. 见: 第三届和谐人机环境联合学术会议(HHME2007). 2007.

[10] Erdem Unal, Elaine Chew, Panayiotis G. Georgiou, et al. Challenging Uncertainty in Query by Humming Systems: A Fingerprinting Approach. in: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval. 2004. 113~118.

[11] Gad M. Landau, U. Vishkin. Efficient string matching with k mismatches. Theoretical Computer Science, 1986, 43(2-3): 239~249.

[12] R. A. Baesa-Yates, C. H. Perleberg. Fast and practical approximate string matching. in: Combinatorial Pattern Matching, Third Annual Symposium. 1992. 185~192.

[13] Z. Galil, R. Giancarlo. Improved string matching with k mismatches. ACM SIGACT News, 1986, 17(4): 52~54.

[14] Ricardo A. Baeza-Yates, G. H. Gonnet. Fast String Matching with Mismatches. Information and Computation, 1994, 108(2): 187~199.

[15] Rodger J. McNab, Lloyd A. Smith, Ian H. Witten, et al. Towards the Digital Music Library:Tune Retrieval from Acoustic Input. in: Proc. ACM Digital Libraries. 1996. 11~18.

[16] Lutz Prechelt, R. Typke. An Interface for Melody Input. ACM Transactions on Computer-Human Interaction, 1998, 8(2): 133~149.

[17] D. Parsons. The directory of tunes and musical themes. Spencer Brown, Cambridge, 1975.

[18] Yunyue Zhu, D. Shasha. Warping Indexes with Envelope Transforms for Query by Humming. in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. 2003.

[19] Roger B. Dannenberg, N. Hu. UNDERSTANDING SEARCH PERFORMANCE IN QUERY-BY-HUMMING SYSTEMS. in: International Conference on Music Information Retrieval(ISMIR). 2004. 205~211.

[20] Roger B. Dannenberg, William P. Birmingham, George Tzanetakis, et al. The MUSART Testbed for Query-By-Humming Evaluation. Computer Music Journal, 2004, 28(2): 34~48.

[21] 王小凤, 周明全, 耿国华, 等. 一个使用歌谱信息进行哼唱检索的系统. 计算机辅助设计与图形学学报, 2007, 19(7).

[22] 魏维, 罗凯, 谢青松. 一种提高哼唱匹配效果的新算法. 黑龙江电子技术, 2007(7).

[23] 马志欣, 付少锋, 周利华. 哼唱检索中一种新的旋律模糊匹配方法. 西安电子科技大学学报, 2006.

[24] 王玮, 蔡莲红, 杨洪涛. 基于内容的音乐检索研究. 见: 第十一届全国多媒体技术学术会议. 2002.

[25] 包先春, 戴礼荣. 哼唱检索系统中一种有效的旋律匹配方法. 计算机仿真, 2008.

[26] 罗凯, 魏维, 谢青松. 哼唱检索中改进的动态时间规整算法. 计算机工程, 2008.

[27] Xiao Wu, Ming Li, Jian Liu, et al. A Top-down Approach to Melody Match in Pitch Contour for Query by Humming. in: International Symposium on Chinese Spoken Language Processing(ISCSLP). 2006.

[28] 袁斌, 许洁萍. 基于 HMM 模型的音乐哼唱检索系统的研究. 见: 第十四届全国多媒体技术、第一届全国普适计算、第一届全国人机交互联合学术会议（第一届全国和谐人机环境联合学术大会）. 2005.

[29] 刘怡, 郝云飞. 大型音乐哼唱检索系统中的近似匹配算法及性能评测. 湖南科技大学学报(自然科学版), 2009(1).

[30] M. M. Association. http://www.midi.org/techspecs/index.php.

[31] 吴宗济, 林茂灿. 实验语音学概要: 高等教育出版社, 1989.

[32] Lawrence R. Rabiner, R. W. Schafer. Digital Processing of Speech Signals. Englewood Cliffs, New Jersey: Pearson Education, 1978.

[33] A. M. Noll. Cepstrum pitch determination. The Journal of the Acoustical Society of America, 1967, 41(2): 293~309.

[34] 郑贵滨, 刘艳, 刘胜, 等. 基于两级神经网络的连续哼唱特征提取. 计算机工程与应用, 2008, 44(18).

[35] E. Barnard, R.A. Cole, M.P. Vea, et al. Pitch detection with a neural-net classifier. IEEE Transactions on Signal Processing, 1991, 39(2): 298~307.

[36] Dan-ning Jiang, Michael Picheny, Y. Qin. Voice-melody Transcription under a Speech Recognition Framework. in: International Conference on Acoustics, Speech, and Signal Processing. 2007.

[37] F. R. Bach, M. I. Jordan. DISCRIMINATIVE TRAINING OF HIDDEN MARKOV MODELS FOR MULTIPLE PITCH TRACKING. in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP). 2005.

[38] Mikel Gainzaf, Bob Lawlor, E. Coylef, et al. Onset Detection and Music Transcription for the Irish Tin Whistle. in: Irish Signals and Systems Conference (ISSC). 2004.

[39] B. Thoshkahna, K.R.Ramakrishnan. AN ONSET DETECTION ALGORITHM FOR QUERY BY HUMMING (QBH) APPLICATIONS USING PSYCHOACOUSTIC KNOWLEDGE. in: European Signal Processing Conference(EUSIPCO). 2009.

[40] Stanley F. Chen, Brian Kingsbury, Lidia Mangu, et al. Advances in Speech Transcription at IBM under the DARPA EARS Program. IEEE Transactions on Audio, Speech and Language Processing, 2006, 14(5).

[41] Tetsuya Shimamura, H. Kobayashi. Weighted Autocorrelation for Pitch Extraction of Noisy Speech. IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, 2001, 9(7).

[42] 张文耀, 许刚, 王裕国. 循环 AMDF 及其语音基音周期估计算法. 电子学报, 2003, 31(6).

[43] 马英, 石小荣, 李海新. 基于 CEP 和 LPC 谱提取语音信号基音周期的方法. 现代电子技术, 2009, 32(20).

[44] Akira Sasou, S. Nakamura. A Pitch Extraction Method Using the Wavelet Transform. Electronics and Communications in Japan, 1999, 82(6).

[45] MIREX. Query by Singing/Humming. http://www.music-ir.org/mirex/wiki/2008:Query_by_Singing/Humming.

致　谢

三年的研究生学习转瞬即逝，让我感觉到时光流逝之迅猛。在语音和语言技术中心实验室里度过了三年，我收获颇多。尽管目前我在某些方面依然不够完善，我自觉三年的研究生活已让我进步很多。首先要感谢导师郑方老师及副导师刘轶老师对我的研究工作的精心指导，郑老师对我的研究工作作了很多深刻而细致的指点，刘老师更是无微不至地关心我的研究，连生活问题也尽力相帮。两位老师既是导师，又是朋友，在此我要向他们说一声“谢谢！您辛苦了！”。

也要感谢实验室的刘建和游展师兄，感谢我们在研究工作中的相互讨论，相互帮助，解开我心中琢磨不透的疑问。还要感谢实验的两位师弟侯珏和张超，我们一起参加了很多研究工作，和师弟们的合作非常愉快。最后要感谢我的家人，感谢他们给予我的理解和关心，感谢他们在我的生活中给予我亲情的温暖，让我倍感幸福。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____日　期：_____

个人简历、在学期间发表的学术论文与研究成果

个人简历

1984 年 12 月 25 日出生于湖南省郴州市。

2003 年 9 月考入清华大学计算机科学与技术系计算机科学与技术专业，2007 年 7 月本科毕业并获得工学学士学位。

2007 年 9 月免试进入清华大学计算机科学与技术系攻读计算机科学与技术硕士至今。

申请的专利

[1] 2009 年，“一种汉语语音识别可信度特征值的计算方法”已申请国家专利，申请号：2008102253536，目前正在审查中。

发表的学术论文

[1] **CAO Wenxiao**, LIU Yi, and Thomas Fang Zheng. Local Mismatch Phone for Confidence Measure in Standard and Accented Chinese Speech Recognition. in: The 6th International Symposium on Chinese Spoken Language Processing(ISCSLP). 2008. 209~212

[2] **曹文晓**, 刘轶, 郑方, 等. 用于哼唱识别精确匹配的线性伸缩动态规划算法. 见: 全国人机语音通讯学术会议(NCMMSC). 2009

[3] **CAO Wenxiao**, LIU Yi, ZHENG Fang, et al. Linear scaling based dynamic programming algorithm for accurate matching in QBH. Journal of Tsinghua University(Science and Technology), 2009, 49(S1): 1402~1407

[4] **Wen-xiao Cao**, Dan-ning Jiang, Jue Hou, et al. A Phrase-level Piecewise Linear Scaling Algorithm for Melody match in Query-by-Humming Systems. in: IEEE International Conference on Multimedia and Expo(ICME). 2009. 942~945