摘　要

语音不仅是人类之间进行信息交流最自然、最有效、最方便的工具，而且也是人与机器之间进行通信的重要工具。语音识别作为一门综合学科，以语音为研究对象，是语音信号处理的一个重要研究方向，它是模式识别的一个分支，涉及到生理学、心理学、语言学、计算机科学以及信号处理等诸多领域。今天，语音识别系统正在不断地改变着我们的生活。而研究噪声环境下的语音识别对于语音识别的应用推广有着极大地现实意义。

本文搭建了一个有限词汇量、非特定人的中文连续语音识别系统，并以此为基础开发了一个车载环境语音识别和控制系统。这套系统的语音识别部分基于剑桥大学开发的语音识别工具——HTK 工具包，软件设计部分则基于微软.NET 架构设计。这套系统包含了车载语音导航、语音拨号、车载设备控制等常见功能，具有一定的现实应用意义。

本文还简要介绍了几种常见的增强语音识别在噪声环境下鲁棒性的方法，并重点研究了基于 SPLICE 算法的非确定性声学解码方法。在实际系统中，采用了 PLP 参数提取特征向量，也获得了一定的抗噪效果。

实验表明，本文基于 HTK 开发的语音识别系统在安静环境下句子识别正确率高达 95.83%。通过采用 PLP 参数特征向量提取的方法，本系统具有一定的抗噪性能。

关键词：语音识别、噪声、HTK

Research on Speech Recognition in the Noise Environment

Abstract

Speech is not only the most natural, effective and convenient tool for human communication, but also an important communication way between human and machine. As a general study, speech recognition bases its research on speech, and is an important research direction for speech signal process. What's more, it's a branch of pattern recognition, too. It refers to physiology, psychology, computer science and signal process etc. Today, speech recognition system is changing our life continuously. So our research on speech recognition in noisy environment has great realistic meaning for its application.

This article sets up a limited-word, speaker-independent, Chinese continuous speech recognition system, and develops a car automatic speech recognition and control system based on it. The speech recognition part of the system is found on the HTK tools, which is developed by Cambridge University. The software part of the system is designed on the .NET framework of Microsoft. This system contains functions such as vehicle navigation, voice dial, and vehicle device control etc, so there is much practical application meaning.

What's more, this article introduces some common methods of strengthening speech recognition's robust for noises, and mainly learns the uncertainty decoding with SPLICE for noise robust speech recognition. In our practical system, we take the PLP parameter as our feature extraction parameter, and it gets some anti-noise effect too.

At last, the experiment shows, our system, based on the HTK tools, reaches its sentence correct rate to 95.83% in the quiet environment. And by taking PLP parameter as our feature extraction parameter, our system has desirable anti-noise ability.

Key Words：Speech Recognition, noise, HTK。

目　　录

引　言

语音识别是指机器通过学习实现从语音信号到文字符号的理解过程，是一种十分重要的人机交互方式。信息产业的迅速发展促使许多研究机构投入了大量的人力、物力和财力来研究语音识别，这一领域的突破也具有重大的现实意义，让机器能够听懂人类的自然语音可以解决诸如智能机器人、语音输入、低码率语音编码等问题，突破信息处理的一个瓶颈。

自 20 世纪 80 年代以来，基于隐马尔科夫模型的语音识别研究方法逐渐成为主流，在安静环境下，语音识别的识别率也达到了一个比较高的水平。比如，剑桥大学的 HTK 大词汇量连续语音识别系统针对非特定人的连续朗读语音识别率已经可以达到 90%以上。在这种情况下，语音识别的鲁棒性研究逐渐成为热点，也成为制约语音识别技术推广和应用的主要因素。

本文研究的范围是噪声环境下的语音识别，综合考虑各研究机构推出的成熟的语音识别系统及其开发工具，本文选择以剑桥大学的 HTK 语音识别工具包为基础作为研究工具。HTK(HMM Tools Kit)是一个剑桥大学开发的专门用于建立和处理 HMM 的实验工具包，主要应用于语音识别领域，也可以应用于语音合成、字符识别和 DNA 排序等领域。为了学习语音识别系统的搭建方法，本文选择搭建一个有限词汇量、非特定人的中文连续语音识别系统；为了研究语音识别在噪声环境下的鲁棒性，本文选择开发车载环境下的语音识别和控制系统。

本文的预期目的是搭建一套具有一定抗噪性能的车载环境下的语音识别和控制系统。并由此总结出搭建语音识别系统的一般方法及常见的抗噪方法。

本文的结构安排如下：首先介绍了语音识别领域在过去几十年里的发展状况；随后详细介绍了语音识别的原理、HTK 的原理、常见的语音识别抗噪技术以及语音合成技术。在第三部分，详细介绍了车载环境语音识别和控制系统的搭建过程，主要是从语音识别系统和软件编写两个角度进行阐述。其中语音识别部分，我们详细介绍了声学模型、语言模型等主要方面。第四部分，我们针对开发的系统，进行了三方面的评估实

验，包括安静环境下的语音识别实验、噪声环境下的语音识别实验以及软件性能的评估分析。

1 研究背景

1.1 本课题国外研究进展

语音识别的研究工作开始于 20 世纪 50 年代，世界上第一个可识别的十个英文数字的语音识别实验系统——Andrey 系统，于 1952 年诞生于美国的 AT&T Bell 实验室。该系统主要依靠测量数字中元音部分的谐波谱识别孤立的数字。

60 年代，计算机的应用推动了语音识别的发展。1960 年，英国的 Denes 等人研究成功了第一个计算机语音识别系统。动态规划（Dynamic Programming，DP）和线性预测分析（Linear Prediction，LP）的提出是这一时期的重要成果，其中后者较好的解决了语音信号产生模型的问题，对语音识别的发展产生了较深远的影响。

70 年代，语音识别领域进一步取得突破。在理论上，LP 技术得到进一步发展，基于 DP 技术的动态时间规整算法（Dynamic Time Warping，DTW）基本成熟，而矢量量化（Vector Quantization，VQ）和隐马尔科夫模型（Hidden Markov Model，HMM）理论的提出则具有里程碑式的重大意义。在实践上，实现了基于线性预测倒谱和 DTW 技术的特定人孤立词语音识别系统。

DARPA(Defense Advanced Research Projects Agency)是在 70 年代由美国国防部远景研究计划局资助的一项 10 年计划，其旨在支持语言理解系统的研究开发工作。CMU（卡内基梅隆大学）、MIT（麻省理工学院）、IBM、AT&T 等都参与了这一计划的开发工作，该计划执行的结果是 1976 年推出了 HARPY(CMU)系统。虽然，这是有限词汇和限定领域的识别系统，但改变了原来只利用声学信息的状况，开始应用高层次语言学知识（如构词、句法、语义、对话背景等）。在这为期 10 年的阶段中尽管所有的研究计划均未能达到预期目标，但它对语音识别和理解研究的发展起了重要的推动作用。通过这一阶段的研究使人们认识到语音识别任务的艰巨性，总结出许多有意义的经验教训，并且从此对语音识别提出了许多基础性的研究课题。这些课题主要涉及到语音信号和自然语言的多变性和复杂性。

80 年代，语音识别研究进一步走向深入，其显著特征是 HMM 模型和人工神经网络（Artificial Neutral Network，ANN）在语音识别中的成功应用。这一时期，隐马尔可夫模型(HMM)技术的成熟和不断完善成为语音识别的主流方法。其次，以知识为基础的语音识别的研究日益受到重视。在进行连续语音识别的时候，除了识别声学信息外，更多地利用各种语言知识，诸如构词、句法、语义、对话背景方面等的知识来帮助进一步对语音作出识别和理解。同时在语音识别研究领域，还产生了基于统计概率的语言模型。同时，人工神经网络在语音识别中的应用研究也逐渐兴起。在这些研究中，大部分采用基于反向传播法（BP 算法）的多层感知网络。人工神经网络具有区分复杂的分类边界的能力，显然它十分有助于模式划分。

80 年代末期由 CMU 推出的 SPHINX 系统，率先突破了语音识别中非特定人、连续语音、大词汇量三大难题，被世界公认为语音识别技术发展中的一个里程碑。

这一时期，美国国防部远景研究计划局又资助了一项为期 10 年的 DARPA 战略计划，其中包括噪声下的语音识别和会话（口语）识别系统，识别任务设定为“（1000 单词）连续语音数据库管理”。到了 90 年代，这一 DARPA 计划仍在持续进行中。其研究重点已转向识别装置中的自然语言处理部分，识别任务设定为“航空旅行信息检索”。

日本也在 1981 年的第五代计算机计划中提出了有关语音识别输入-输出自然语言的宏伟目标，虽然没能实现预期目标，但是有关语音识别技术的研究有了大幅度的加强和进展。1987 年起，日本又拟出新的国家项目——高级人机口语接口和自动电话翻译系统。

进入 90 年代以来，在语音识别的系统框架方面并没有什么重大突破。但是，在语音识别技术的应用及产品化方面出现了很大的进展。随着多媒体时代的来临，一个巨大的人机语音交互市场正在形成，这迫切要求语音识别系统从实验室走向使用。许多发达国家如美国、日本、韩国以及 IBM、微软、Intel、AT&T 等著名公司都为语音识别系统的实用化研究投入巨资。同时，随着语音识别各个方面问题的逐个解决，语音识别中最困难的非特定人、大词汇量、连续语音识别已经达到了较高的性能。各研究机构推出的

识别系统包括：IBM 的 ViaVoice 系统，Microsoft 的 Whisper 系统等。还有剑桥大学开发的 HMM 工具包，它已经成为研究人员研究语音识别的重要工具。[1]

噪声环境下的语音识别鲁棒性研究，特别是现实环境噪声对说话人识别性能的影响，是当前研究的重点。目前世界上许多大学著名的大学、研究机构以及很多的大公司的实验室都在进行语音识别鲁棒性的研究。比如麻省理工学院林肯实验室的鲁棒语音识别组、NTT 的 Furui 研究所，俄勒冈研究生院（OGI）的 Hermansky 教授及其领导的人类信号处理小组，SRI 公司的语音技术与研究实验室（STAT），瑞士的 Dalle Molle 感知人工智能研究所（IDIAP）等等。国际声学、语音与信号处理会议也专门设有语音识别鲁棒性研究专题。

1.2 本课题国内研究进展

我国的语音识别研究起始于 1958 年，由中国科学院声学所利用电子管电路识别 10 个元音。直至 1973 年才由中国科学院声学所开始计算机语音识别。由于当时条件的限制，我国的语音识别研究工作一直处于比较缓慢的发展阶段。

进入 80 年代以后，随着计算机应用技术在我国逐渐普及和应用以及数字信号技术的进一步发展，国内许多单位具备了研究语音技术的基本条件。与此同时，国际上语音识别技术在经过了多年的沉寂之后重又成为研究的热点，开始发展迅速。就在这种形式下，国内许多单位纷纷投入到这项研究工作中去。

1986 年 3 月我国高科技发展计划(863 计划)启动，语音识别作为智能计算机系统研究的一个重要组成部分而被专门列为研究课题。在 863 计划的支持下，我国开始了有组织的语音识别技术的研究，并决定了每隔两年召开一次语音识别的专题会议。从此我国的语音识别技术进入了一个前所未有的发展阶段。

目前，国内从事语音识别研究的单位超过了几十个：如清华大学、中科院自动化所、声学所、深圳先进技术研究院、哈尔滨工业大学、四川大学等。它们结合汉语语音学和语言学的特点，在基础理论、模型和使用系统等方面做了大量工作，并取得了较好的成果。台湾在汉语语音识别方面的研究也具有较高水平，其中以 Lin-Shan Lee 教授主持的研究小组最为出色，它们研制成功了一个实时汉语语音听写机——Golden Mandarin。

特别值得一提的是 IBM 公司开发的 ViaVoice 汉语语音识别软件，它代表了汉语语音识别较高水平。[2]

国内开展语音识别鲁棒性研究工作相对比较晚，但也已经引起了广泛的关注，许多大学和研究机构都在进行这一领域的研究。其中比较有代表性的有北京大学视觉与听觉处理国家重点实验室；中国科学院自动化研究所模式识别国家重点实验室、中科院声学所、清华大学、国防科技大学、上海交通大学、北京邮电大学等也都取得了丰硕的成果。此外，微软亚洲研究院语音组、中科院声学所的中科信利技术有限公司、中科院自动化所的模识科技公司等许多高科技公司也都为增强语音识别的鲁棒性做出了有意义的尝试。

2 理论基础

2.1 课题背景及开展研究的意义

语言是人类最重要的一种交流方式，也是人类最方便、最自然、最理想的表达方式；让机器听懂人类的语音并为人类服务，以及让机器发出人类语音，最终实现和人类交流，一直是人类的理想之一。这些都促使人们不断去探索语音识别技术。

随着计算机性能的不断提高和语音识别技术的不断发展，当前对于安静环境下的语音识别技术已经相当成熟，识别的准确率也达到了较高的水平，然而环境的多样性特别是噪声的多样性给当前的语音识别系统造成了很大的干扰，使得现有的系统的识别率急剧下降，也使得语音识别系统的实用性和推广性受到很大的程度的影响。因此，本文研究噪声环境下的语音识别是具有极大意义的。

2.2 语音识别的原理

语音识别基本构造如 Fig1 所示，系统可分为前端处理和后端处理，前端处理包括语音的录入、处理、特征值的提取，后端则是个跨数据库的搜索过程，分为训练和识别两个部分，训练是对所建的模型进行评估、匹配、优化，获得模型参数，识别是一个专用的搜索数据库，获取前端数据数值后，在声学模型、一个语言模型和一个字典下共同完成。声学模型表示一种语言的发音声音，可以通过训练来识别特定用户的语音模型和发音环境的特征，语言模型是对语料库单词规则化的概率模型。字典列出了大量单词及发音规则。总体来说语音识别是一个模式识别匹配的过程。在这个过程中，计算机首先要根据人的语音特点建立声学模型，对输入的语音信号进行分析，并抽取所需的特征，在此基础上建立语音识别所需的模板。然后，在识别过程中，计算机根据语音识别的整体模型，将计算机中已经存有的语音模板与输入语音信号的特征进行比较，并根据一定的搜索和匹配策略找出一系列最优的与输入语音匹配的模板。最后通过查表和判决算法给出识别结果。显然，识别结果与语音特征的选择、语音模型和语言模型的好坏、模板是否准确等都有直接的关系。[3]

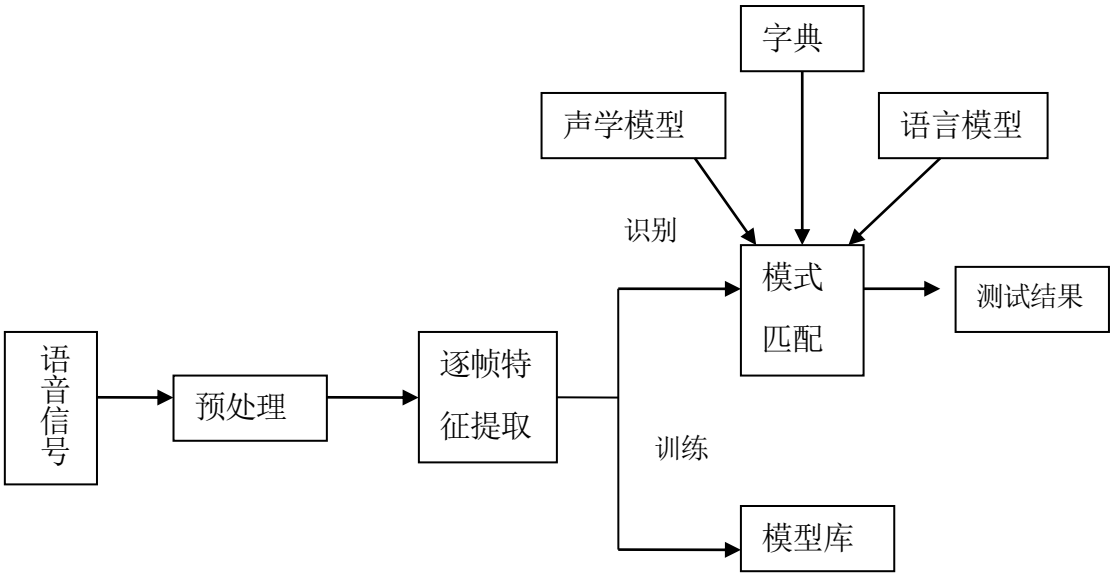

Fig1 语音识别系统

2.3 HTK 原理简介

2.3.1 综述

HTK（HMM Tools Kit）是英国剑桥大学开发的专门用于建立和处理 HMM 的实验工具包，主要应用于语音识别领域，也可以应用于语音合成、字符识别和 DNA 排序等领域。另外，HTK 还是一套源代码开放的工具箱，其基于 ANSI C 的模块化设计方式可以方便的嵌入到用户系统中。[4]

HTK 是一个基于隐性的马尔科夫链模型（HMM）的工具包，可用来建立以 HMM 为基础的语音识别系统。建立这样的语音识别系统一般包含了两个过程：一是通过训练来获得所建立的语音模型的相关参数；二是应用这些参数，从而利用 HTK 来识别新的语音信息。[5]

HTK 整个处理数据的过程如 Fig2 所示，对于里面每个过程在本文第三部分会有详细介绍。

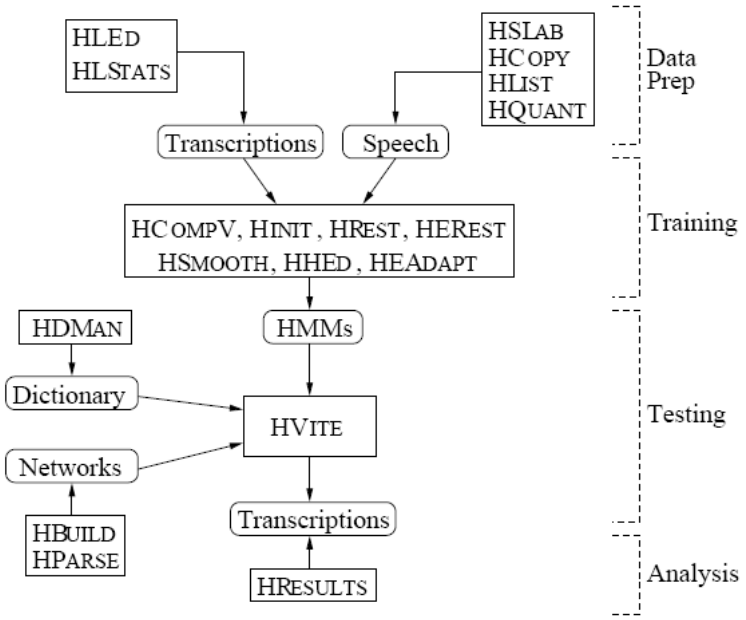

Fig2 HTK 处理过程示意图

2.3.2 孤立词汇语音识别

语音是一个随机过程，每次发音时，我们都可以得到一个帧矢量序列（称为观察序列）

$$O: O = \{o_1, o_2, ..., o_T\}$$

对于同一词的不同发音，O 的帧数 T 和o_i都在变化，可以看成是该随机过程模型的多次实现。从语音产生的过程来看，可以想象为声道沿不同位置转移时，每一位置产生一随机声学输出。可以把各声道位置想象成各个状态S_i，而发现序列可想象为在该状态下的一个随机输出o_i。因此，语音的随机过程可以看成是两个随机过程构成的：

一是状态转移的随机过程；二是输出的随机过程。

在对信号的建模问题上，隐马尔科夫模型（HMM）既解决了用短时模型描述平稳段的问题，又解决了每一个短时平稳段是如何转变到下一个短时平稳段的问题。它由相互关联的两个随机过程共同描述信号的统计特性，其中一个是隐蔽的(不可观测的)具有有限状态的马尔可夫链，另一个是与马尔可夫链的每一状态相关联的观察矢量的随机过程(可观测的）。隐马尔可夫链的特征要靠可观测到的信号特征揭示。这样，语音等时变信号某一段的特征就由对应状态观察符号的随机过程描述，而信号

随时间的变化由隐马尔可夫链的转移概率描述。故HMM很好的描述了语音生成的这两个随机过程。

如果用随机过程$W:W=w_1,w_2,...,w_s$表示一串词汇，语音识别的目的是要计算$\arg\max_i\{P(w_i\mid O)\}$，即通过语音识别器选择一个在给定发音情况下最大似然的单词串作为识别结果。根据贝叶斯公式，有：

$$P(w_i\mid O)=\frac{P(O\mid w_i)P(w_i)}{P(O)},$$

而$P(O)$是观察到的特征矢量序列的概率，总是已知的，和识别过程无关；而$P(w_i)$表示一个合法的单词序列 w 出现的概率，即语言模型，可以通过一个纯的文本序列语料库来训练这个模型。在孤立的语音识别中，我们认为 w 中每个词$(w_1,w_2,w_3,...,w_s)$都是独立的，因此$P(w_i)$与识别过程无关。但是在连续的语音识别中，需要为$P(w_i)$预先建模。

对于孤立词汇的语音识别系统，只需要计算出$P(O\mid w_i)$，这就需要通过建立 HMM（隐马尔科夫模型）来建立声学模型求得。[5]

2.3.3 HMM 的三个经典算法

对于给定的一个观察序列$O=o_1o_2...o_r$和一个 HMM 参数组$\lambda=(\pi,A,B)$：

前向后向算法（Forward Backward Procedure），用来概率$P(O\mid\lambda)$的计算，这实际上是一个模型评估的问题，因为$P(O\mid\lambda)$反映了观察序列和与模型的吻合程度。Viterbi 算法用来确定最佳状态链$Q=q_1q_2...q_T$，从而解释序列O。Baum-Welch 算法则用来进行 HMM 参数优化。

2.3.4 连续语音识别

简单的说，连续语音识别（continuous speech recognition）只需要把孤立词汇语音识别所建立起来的 HMM 连接起来便可实现。每个基本语音单元对应着一个特定的隐马尔科夫模型（HMM），将这一系列的 HMM 链接起来构成一个组合的 HMM 对应着整个句子。由此我们可以通过这个组合 HMM 计算 P(O|W)。另外，根据语言模型可以计算

P(W)。最终，我们依据不同词序列 W 的 P(O|W)P(W)选择最有可能的词序列作为连续语音识别系统的输出。

2.4 常见的抗噪方法简介

目前纯净语音识别已达到一个比较成熟的阶段，以 IBM 的 Via Voice 为代表，其对连续语音的识别率可以达到 95%以上，但是对语音输入环境有较严格的要求，否则系统性能将会有很大的下降。造成这种情况的原因是训练环境和识别环境的差异造成了模型和测试数据之间的失配。现在很多识别系统的参数都是在实验室环境中通过训练得到的，训练语音大多是在安静的情况下，通过高质量麦克风采集的。但是，在实际应用场合，由于多种因素的影响，待识别语音不可避免的会和系统参数存在失配，从而造成实际性能和实验室中的性能的巨大差别。因此，在实际的系统上，很有必要让大词汇量语音识别系统加强对干扰噪声的鲁棒性。现在很多实现抗噪的方法已经被研究出来了，归纳起来主要有以下两类：[6]

2.4.1 基于前端处理（特征提取）的抗噪

基于特征的抗噪识别技术已经发展了多年，比较成功并广泛应用的有感知线性预测参数(perceptual linear predictive, PLP)，RASTA-PLP 及其扩展参数，倒谱均值规整（cepstral normalization）和 SPLICE[7]等等。

PLP(Perceptual Linear Predictive)感知线性预测技术是模仿人耳听觉特性来提取语音信号鲁棒特征的方法。该参数模型主要在三个层次上模仿了人的听觉感知机理：（1）临界频带分析处理；（2）等响曲线预加重；（3）信号强度-听觉响度变换。从人耳的掩蔽效应的原理出发，不仅考虑了临界带宽这种特性，并且考虑到耳蜗的分频特性，另外它有计算量小、维数低的特点。[8]具体的流程图如 Fig3 所示：

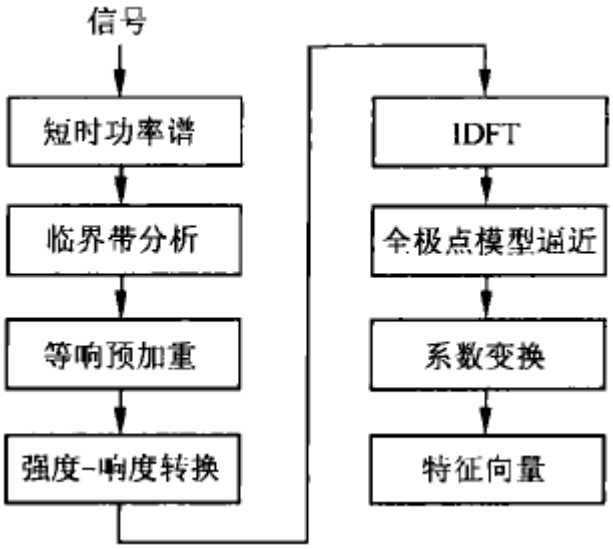

Fig3 PLP 算法实现图

PLP 方法能够很好的模拟人耳的听觉特性，提取出鲁棒的体现噪声目标信号特征的向量，分类效果明显，提取的特征维数较低，便于实时处理。

RASTA(RelAtive SpecTrA)RASTA 方法采用带通的滤波器加权倒谱特征，抑制低阶和高阶分量。因为背景噪声和语音信号通过全极点滤波器后的输出信号平均倒谱的方差比显示，高频倒谱分量更容易受到噪声的影响，因此在进行频谱相似度比较时的可靠性低。而开始的几个低频分量的变化主要受信道特性，说话人特性和声道的影响，对于非特定人的语音识别系统，这些分量也是不利的。所以采用带通滤波器强调倒谱中间部分在识别中的作用，可以提高系统的稳健性。[6]

倒谱均值归一化(Cepstral Mean Normalization,CMN)CMN 是语音识别系统经常采用的一种稳健算法，其目标是为了消除电话信道等卷积噪声在倒谱域造成的偏差，但对背景噪声也非常有效。[6]

SPLICE(Stereo-based Piecewise Linear Compensation for Environment) SPLICE 可以看作是特征空间的 MMSE 增强算法。SPLICE 算法假设噪声对语音概率分布均值的影响是线性的，从而得到简化的 MMSE 估计结果。SPLICE 中事先训练一系列噪声模型来存储各种不同的噪声环境的信息，识别时通过 MAP 估计最接近的噪声模型，然后结合此模

型的先验信息来获得噪声参数的估计。对于 SPLICE 的详细介绍请参阅附录中的外文译文部分。

2.4.2 基于模型的抗噪

基于模型的抗噪识别方法直接估计与噪声环境相匹配的声学模型参数以减小系统训练与测试情况下的差异。研究表明，此类方法性能优于基于特征的技术，因此已成为二十年来的抗噪鲁棒识别的主流技术。当中最主要的方法包括并行模型组合 parallel model combination (PMC)[9]，基于泰勒级数展开的近似 Vector Taylor Series (VTS) expansion[10,11], 以及基于模型的非确定性声学解码 Uncertainty Decoding (UD)[12]。

并行模型组合(Parallel Model Combination,PMC) PMC 算法是模型空间最经典的算法，其首先利用采集到的噪声数据训练噪声模型，然后将噪声模型和语音模型在对数谱域进行合并，合并的过程就是对概率密度函数的补偿，包括对均值和方差的调整。PMC 算法中分析噪声对模型影响的理论框架和实际吻合得非常好，抓住了问题的关键，所以其对语音识别系统抗噪声性能有显著的提高。[6]

泰勒级数分析法(Vector Taylor Series,VTS) VTS 算法将噪声对语音特征的影响通过泰勒级数在纯净语音特征均值附近展开，并选取线性项作为近似。纯净语音的特征将通过 MMSE 估计的方式按照上述近似方程给出，从而消除噪声对特征带来的偏差。[6]

2.5 语音合成技术

当前，语音合成的研究已经进入到文字-语音转换（TTS）阶段，其功能模块可分为文本分析、韵律建模和语音合成三大模块。其中，语音合成是 TTS 系统中最基本、最重要的模块。概括起来说，语音合成的主要功能是：根据韵律建模的结果，从原始语音库中取出相应的语音基元，利用特定的语音合成技术对语音基元进行韵律特性的调整和修改，最终合成出符合要求的语音。

语音合成技术经历了一个逐步发展的过程，从参数合并到拼接合成，再到两者的逐步结合，其不断发展的动力是人们认知水平和需求的不断提高。目前，常用的语音合成技术主要有：共振峰合成、LPC 合成、PSOLA 拼接合成和 LMA 声道模型技术。它们各有优缺点，人们在应用过程中往往将多种技术有机地结合在一起，或将一种技术的优点运用到另一种技术上，以克服另一种技术的不足。[13]

3 实验方法

3.1 实验系统搭建背景

为了尽可能方便的对语音识别技术特别是噪声环境下语音识别展开研究，经过仔细思考，决定设计一个车载环境语音识别和控制系统。该系统作为一个有限词汇量的非特定人连续语音识别系统，涵盖了语音识别的各个过程和阶段，对于全面总体认识语音识别技术有极大的促进作用；另一方面，该系统选择在车载环境下，本身就是一个带有特定噪声的环境，因此非常切合本课题——噪声环境下的语音识别。为了便于演示，控制系统部分选择在 PC 上模拟车载设备的控制效果。

3.2 实验系统需求分析

车载环境的语音识别和控制系统的功能主要包括以下几个部分：

①语音导航

例如，司机可以询问“从海上世界到先进院怎么走”，系统会给出相应的导航路线和导航地图；也可以查询当前位置。

②语音拨号

例如，司机可以说“拨打电话 110”，系统会模拟电话拨号；

③车载设备控制

包括空调控制、空调温度调节、车载 CD、收音机控制、雨刷、车灯、座椅等的控制；

④车身信息查询

包括当前车速、当前油量的语音查询。

⑤识别结果语音提示

系统在识别出语音后，会合成相应语音进行提示。

3.3 实验系统架构设计

3.3.1 开发环境

本系统选择在.NET Framework 2.0 基础上开发，开发环境为 Visual Studio 2008，开发语言为 C#。系统采用分层架构，基于组件编程的理念，最大化的实现软件工程理论中代码复用的核心思想。

3.3.2 功能结构分析

车载环境语音识别和控制系统分为识别系统和控制系统两个部分，如 Fig4 所示：

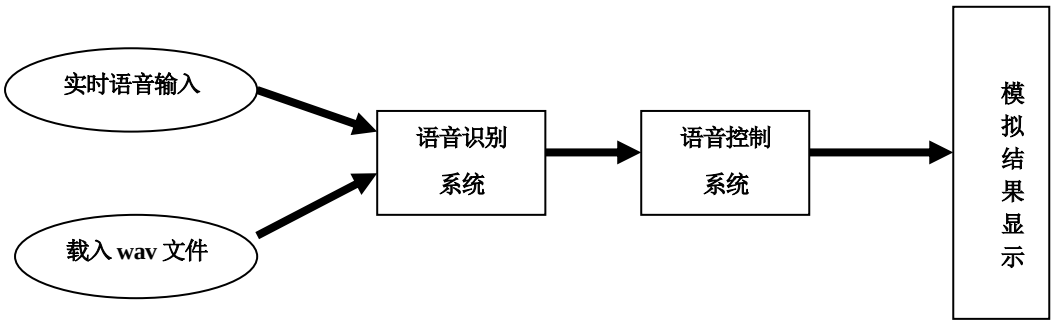

Fig4 车载环境语音识别和控制系统功能结构图

其中输入端可以是实时的音频信号，也可以是录好的 wav 格式的音频文件，输入端的信号输入后，被送入识别系统，进行特征提取、识别，最后输出识别结果；控制部分从识别结果中提取有用的识别文本信息，并且进行命令匹配，生成相应的控制指令；最后，系统根据返回的控制指令，完成车载设备的模拟控制效果。

3.3.3 系统架构

根据系统功能结构的分析，系统架构应该如 Fig5 所示：

其中 CarASR_UserInterface 为界面表现层，主要负责与用户进行交互以及模拟结果的展示，在本系统中由主界面和载入文件界面组成；

CarASR_Control 为控制层，对应着系统的控制系统部分，主要完成读取识别文本，生成控制指令；

CarASR_Recognition 为识别层，对应着系统的识别系统部分，主要完成对语音信号的特征提取和识别；

CarASR_DataAccess 为数据访问层，主要用来访问数据库，控制层在完成语音导航功能时需要检索数据库，查询相应的导航信息；

底层则为 The HTK Tools，本系统使用的是 3.4 版本，系统会调用 HTK Tools 中的 Hcopy 和 Hvite 命令来完成相应功能；.NET Framework 2.0 则为程序提供统一语言运行时和相应的组件。

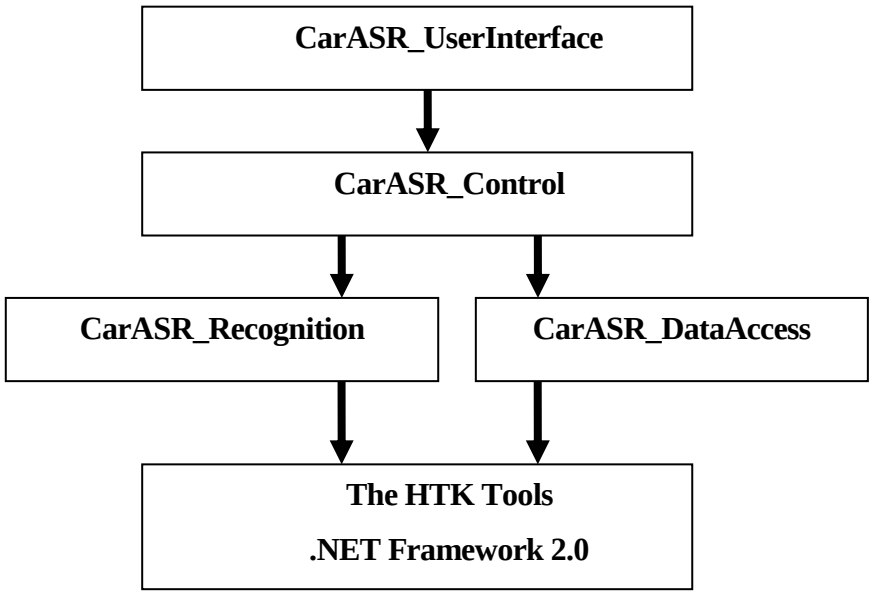

Fig5 车载环境语音识别和控制系统架构图

3.4 实验系统实现方法

3.4.1 基于 HTK 的语音识别

3.4.1.1 数据采集

在本项目中，我们设计并采集了真实的说话人中文、连续语音数据。这种中文语音和脚本（orthography transcription）数据更具有实效性和推广价值。该语音数据库的设计和采集包括以下几个方面：

①文本内容的设计：针对驾驶环境下的语音内容，将设计若干在以下几个方面的话题，包括各种控制命令；地名、数字等常用信息；含音素丰富的句子和词语。其中，汽车控制命令（含导航控制命令）是文本内容设计的重点。

如表 1 所示，数据采集中共采集了包括数字、地名、控制指令以及随机控制短语在内的 101 个词或短语，详细的内容请参见附录——语音数据库的设计。

表 1 文本内容表

类别	内容示例
数字 单独 10 个数字（共 15 个） 随机电话号码（共 5 个） 随机温度值（共 5 个）	例如：一、幺、十、二十… 例如：幺三八四零九六 例如：十三、二十六
地名 常用的地名 （共 45 个）	例如：世界之窗、海上世界、先进院
控制指令 （共 25 个） 导航（6 个） 通信（1 个） 娱乐（7 个） 车辆信息（2 个） 辅助设备（9 个）	例如：打开导航信息、从、到 例如：拨打电话 例如：播放 CD、收音机 例如：显示油量、显示车速 例如：打开雨刷、抬高座椅
随机控制短语	

导航（6 个）	例如：从海上世界到先进院怎么走
通信（5 个）	例如：拨打电话一三八五四六七九
辅助设备（5 个）	例如：温度设定二十四、温度设定十六

②录音人员：共 30~40 人。考虑驾驶者男女比例音素，计划男女比例为 3:2。考虑驾驶者年龄构成情况，录音者年龄 20-50 岁之间。考虑到口音问题，尽量聘请不同地区驾驶者进行数据采集工作。

在实验中，我们随机邀请到了中国科学院深圳先进技术研究院的共计 33 名学生，其中女性 14 人，占 42.4%，男性 19 人，占 57.6%。年龄最小的为 18 岁，年龄最大的为 27 岁。根据语音数据库的设计，采集每个人对语音数据库中 101 个词或句子的语音数据。详细的人员情况如表 2 所示：

表 2 录音人员地域分布

口音	人数	口音	人数	口音	人数
北京	1	湖南	4	山西	2
辽宁	2	河北	2	江西	5
山东	1	重庆	1	吉林	1
河南	1	湖北	6	广东	1
福建	1	四川	1	浙江	1
云南	1	安徽	1	江苏	1

经过统计，所有用来训练的语音数据总时长约为 8.25 小时，用来测试的语音数据总时长约为 0.75 小时。

3.4.1.2 构造语言模型

语言模型（Language Model，LM）是为了在语音识别的过程中有效地结合语法和语义的知识，提高识别率，减少搜索的范围。由于很难准确的确定词的边界，以及声学模型描述语音变异性的能力有限，识别时将产生很多概率得分相似的词的序列。因此，

在实用的语音识别系统中通常使用语言模型 P(W)从诸多候选结果中选择最有可能的词序列来弥补声学模型的不足。

在本系统中，为了尽可能提高识别率，我们将常用的指令作为孤立的词汇写入词汇网络，例如“打开空调”作为一个整体存在于字典中。对于导航指令和拨打电话指令以及辅助设备指令（空调温度设定）定义了如 Fig6 所示的语法规则：

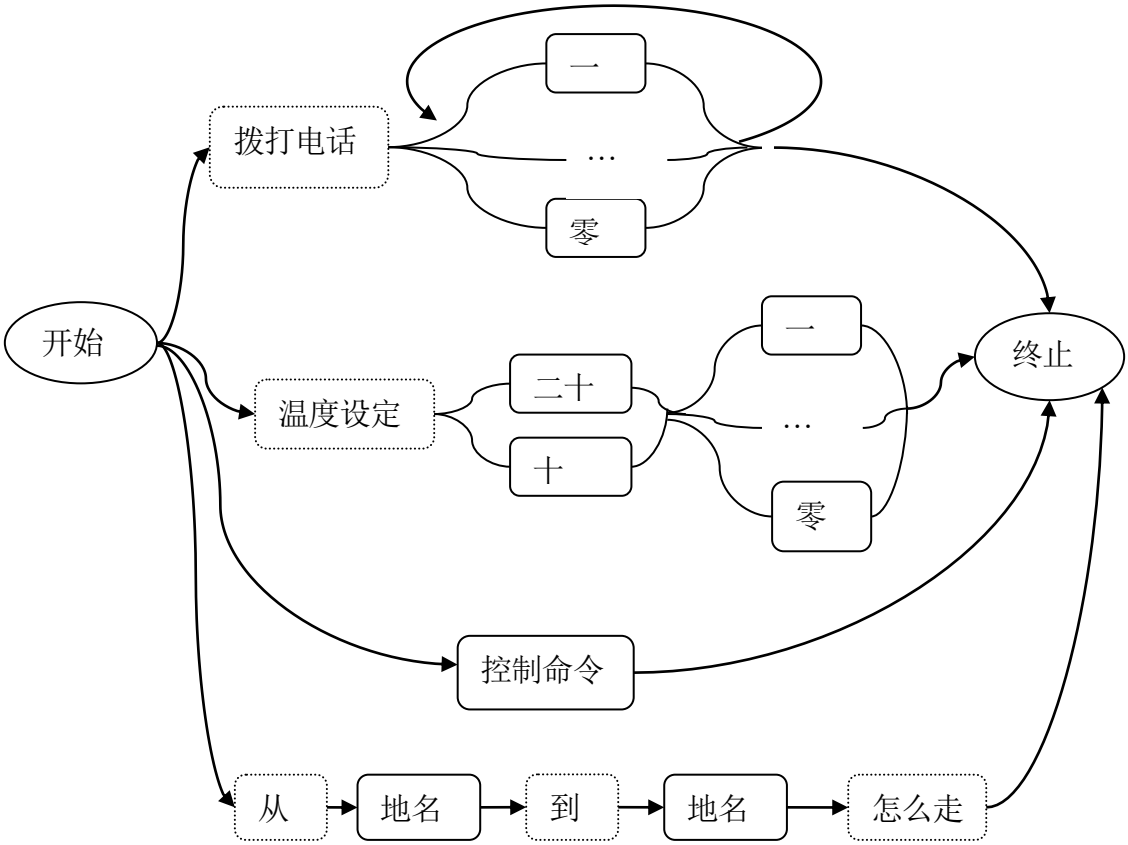

Fig6 词网络图

3.4.1.3 **字典和音素单元**

字典记录的是词的标准发音，用来作为音素和识别文本的对照标准。在本系统中，由于是出现的词汇是有限的，因此选择了手工编写字典。并且为了提高识别率，将常用的控制指令作为一个专有名词进行标注，例如“打开空调”的出现在字典中为“d a3 k ai1 k o1 ng t iao2”。

此外，字典中所使用音素集合为带音调（pitch）的音素单元，比如“打”的发音在字典中表示为“打 d a 3”。我们的声学模型也是根据上述音素单元进行建模的。用于声学模型的音素单元表如表 3 所示：

表 3 用于声学模型的音素单元表

	音素单元
音素I	b;c;ch;d;f;g;h;j;k;l;m;n;ng;p;q;r; s;sh;t;w;x;y;z;zh

音素II	a(1-5);ai(1-5);ao(1-5);e(1-5);ei(1-5);
	er(1-5);i(1-5);ia(1-5);iao(1-5);ie(1-5);
	iu(1-5);o(1-5);ou(1-5);u(1-5);ua(1-5);
	uai(1-5);ue(1-5);ui(1-5);uo(1-5);v(1-5)

3.4.1.4 **特征提取**（HCopy）

这是数据准备阶段的最后一个步骤，目的是将音频信号波形进行编码，提取相应的特征向量。在信号处理系统中，对采样语音信号进行预处理是必要的，这样可以保证系统获得一个比较理想的处理对象。在语音信号处理中，常用的特征参数模型包括 Mel 倒谱系数（Mel Frequency Cepstrum Coefficient,MFCC）和感知线性预测系数（Perceptual Linear Predictive,PLP）。在 HTK 中，HCopy 工具即是用来特征参数提取的，支持以上两种特征参数的提取。

如 Fig7 所示，在 HTK 中，HCopy 命令使用如下：

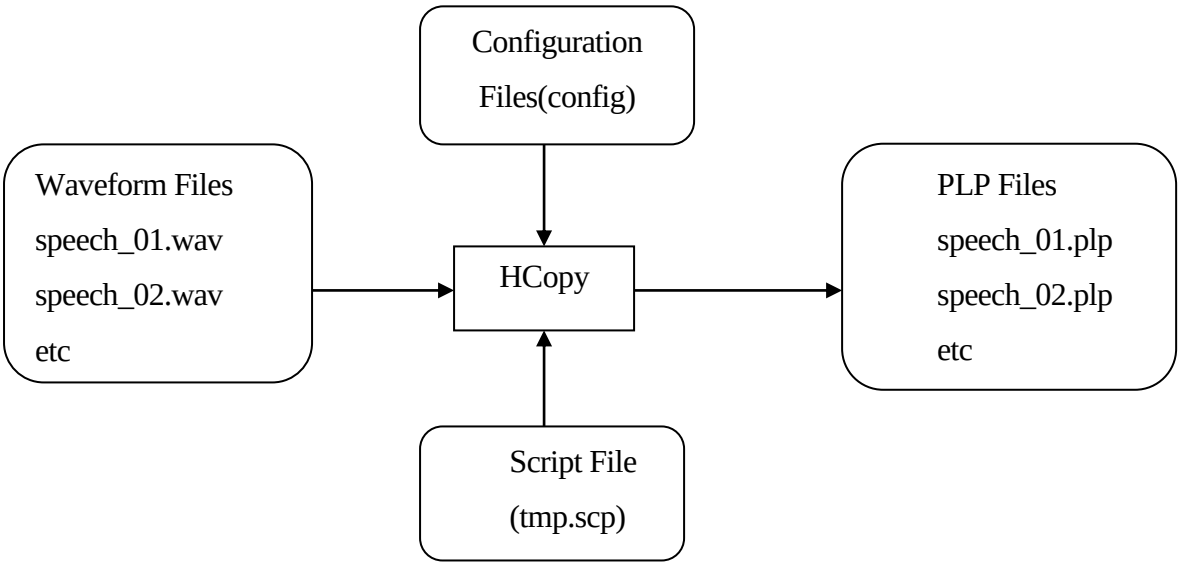

Fig7 HCopy 用法示意图

需要 wav 格式的音频数据文件；需要 config 文件，即配置文件；还需要 scp 文件，即脚本文件，记录的是输入 wav 文件和输出 plp 文件的对应关系。通过调用 HCopy 命令，就可以生成相应的 plp 文件了。

其中配置文件格式一般如下：

```
SOURCEFORMAT = WAV          # Gives the format of the speech files
TARGETKIND = MFCC_0_D_A     # Identifier of the coefficients to use
# Unit = 0.1 micro-second :
WINDOWSIZE = 250000.0       # = 25 ms = length of a time frame
TARGETRATE = 100000.0       # = 10 ms = frame periodicity
NUMCEPS = 12                # Number of MFCC coeffs (here from c1 to c12)
USEHAMMING = T              # Use of Hamming function forwindowing frames
PREEMCOEF = 0.97            # Pre-emphasis coefficient
NUMCHANS = 26               # Number of filterbank channels
CEPLIFTER = 22              # Length of cepstral liftering
```

其中 SOURCEFORMAT = WAV 表示待处理的是 wav 文件，TARGETKIND = MFCC_0_D_A 表示转化为 mfcc 文件。NUMCEPS = 12 表示使用的是 12 维的 Mel 倒谱系数；USEHAMMING=T 表示使用汉明窗；PREEMCOEF = 0.97 表示预加重系数为 0.97； NUMCEPS、NUMCHANS、CEPLIFTER 都是 MFCC 计算中常用到得变量。Windows Size 和 Frame Rate 是相互独立的。通常来说，Windows Size 会比 Frame Rate 大，这样才能保证窗口的重叠，如 Fig8 所示：

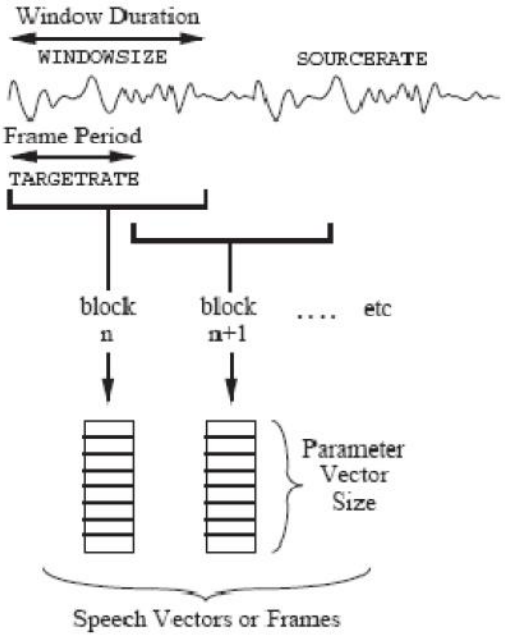

Fig8 提取特征示意图

在本实验中，为了使系统具备抗噪能力，采用的是具有抗噪能力的感知线性预测系数做特征提取参数，所以配置文件应写成 TARGETKIND = PLP_0_E，表示输出的是 PLP 文件。

3.4.1.5 构造声学模型

声学模型是识别系统的底层模型，也就是要对 P(O|W)建模。声学模型是语音识别系统中最为关键的一部分。声学模型的目标是提供一种有效的方法，计算语音的特征矢量序列和每个发音模版之间的距离。

整个训练分为两个阶段，即单因子模型的训练和三因子模型的训练。每个阶段又都包括了循环往复的两个步骤，即估计与重对齐。单因子模型应该作为模型训练的出发点，然后才是三因子模型的训练。为了在识别时预测那些未在训练集中出现过的三因子，需要对状态被捆绑在一起的三因子模型的参数进行估计。

在实验中，首先我们针对音素单元训练了单音素（monophone）模型，之后，我们考虑到汉语的标注是一种字典式标注而不是发音标注，因此将每个语音波形与其标注进

行了强制对齐，并且允许在音素之间插入或删除短时停顿（sp），以及在每句（utterance）的开始和结尾处插入静音（sil）。

在初始单音素模型训练之后，我们在全音素字典的基础上，针对上下文关系，将单音素扩展成了所有可能的三音素（triphone）。我们采用决策树对扩展后的三音素进行了绑定。

在 HTK 中，HERest 工具可以用来创建单音素（Monophone）语音模型以及创建捆绑状态（Tied-state）的三音素（Triphones）声学模型。

3.4.1.6 识别过程（HVite）

对于连续语音识别来讲，识别的最终目的是从各种可能的因素模型状态序列形成的网络中找出最优的词序列（即最优路径）。这实质上属于解码算法或搜索算法的范畴。

语音识别算法的具体识别搜索算法的实现要根据语言的特点、模型的整体结构进行设计。语音识别的搜索算法可以分为两类。其中，第一类称为深度优先（Depth-first）；第二类称为宽度优先（Breadth-first）。深度优先的算法通常包括堆栈解码器、A*解码算法。宽度优先算法使用维特比（Viterbi）解码算法。这两种算法各有优缺点。当精确的向前预测信息可以获得，深度优先算法具有计算量和存储量小的优点，而宽度优先算法具有帧同步计算的优点，但计算量较大。[16]

HTK 中的 HVite 命令为一个采用 Viterbi 算法的识别器，它会将语音信号和一系列隐马尔科夫模型相匹配，然后对应着每个信号输出一个包含识别结果的文本文件。其识别过程如 Fig9 所示：

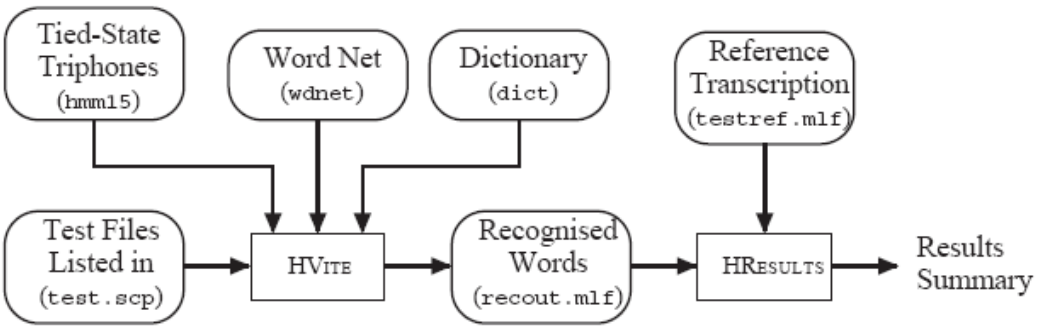

Fig9 HVite 用法示意图

其中 hmm15.mmf 文件指的是最终训练好的绑定状态的三音素 HMMs，即声学模型；wdnet.slf 指的是词网络，用来描述可能出现的词汇之间的语法形式，即语言模型；dict 指的是字典文件，用来记录每个词的标准发音，test.scp 文件指的脚本文件，里面记录着需识别的语音经过编码后的特征向量文件列表。通过 HVite 命令，将识别的结果输出为 recout.mlf 文件了。

3.4.2 基于.NET 的软件设计

3.4.2.1 基于 DirectSound 的录音程序

DirectSound 是微软公司 DirectX 中的一个组件，软件开发者可将数据通过多个音频流直接写入任何 DirectX 兼容声卡里。它支持多种采样频率，且能随意增加以软件为基础的声音特效。同时，DirectSound 本身就是一个声音合成引擎，它用系统内存容纳不同的音频流。

为了实现一个录音的基本过程，至少需要以下对象的支持：一是录音设备，对 PC 而言也就是声卡。这个录音设备可以进行的操作应该有开始和关闭。二是缓冲区，也就是录制的声音放在哪里的问题。

DirectSound 对录音的支持类如下：

Capture，设备对象，可以看作是声卡的描述。

CaptureBuffer，缓冲区对象，存放录入的音频数据。

Notify，事件通知对象，由于录音是一个长时间的过程，因此使用一个缓冲队列（多个缓冲区）接收数据，每当一个缓冲区满的时候，系统使用这个对象通知应用程序取走这个缓冲区，并继续录音。

以上三个对象是进行录音操作的主要对象，本系统是使用 C#的托管代码（ Managed Code），除了以上三个主要的 DirectSound 类，还需要以下几个辅助类。

WaveFormat，描述了进行录制的声音波形的格式，例如采样率，单声道还是立体声，每个采样点的长度等等。

Thread，线程类，由于录音的过程是需要不断处理缓冲区满的事件，因此新建一个线程对此进行单独处理。

AutoResetEvent，通知的事件，当缓冲区满的时候，使用该事件作为通知事件。

在本系统中，引用了 Microsoft.DirectX.DirectSound.dll 和 Microsoft.DirectX.dll 两个动态链接库，完成了录音程序的开发，详情可以查阅源码的 recorder 类。

3.4.2.2 非托管动态链接库（unmanaged dll）的调用

动态链接库（也称为 DLL，即为“Dynamic Link Library”的缩写）是 Microsoft Windows 最重要的组成要素之一，打开 Windows 系统文件夹，你会发现文件夹中有很多 DLL 文件，Windows 就是将一些主要的系统功能以 DLL 模块的形式实现。

动态链接库是不能直接执行的，也不能接收消息，它只是一个独立的文件，其中包含能被程序或其它 DLL 调用来完成一定操作的函数(方法。注：C#中一般称为“方法”)，但这些函数不是执行程序本身的一部分，而是根据进程的需要按需载入，此时才能发挥作用。

DLL 只有在应用程序需要时才被系统加载到进程的虚拟空间中，成为调用进程的一部分，此时该 DLL 也只能被该进程的线程访问，它的句柄可以被调用进程所使用，而调用进程的句柄也可以被该 DLL 所使用。在内存中，一个 DLL 只有一个实例，且它的编制与具体的编程语言和编译器都没有关系，所以可以通过 DLL 来实现混合语言编程。DLL 函数中的代码所创建的任何对象（包括变量）都归调用它的线程或进程所有。

下面列出了当程序使用 DLL 时提供的一些优点：

1)使用较少的资源。当多个程序使用同一个函数库时，DLL 可以减少在磁盘和物理内存中加载的代码的重复量。这不仅可以大大影响在前台运行的程序，而且可以大大影响其他在 Windows 操作系统上运行的程序。

2) 推广模块式体系结构。DLL 有助于促进模块式程序的开发。这可以帮助开发要求提供多个语言版本的大型程序或要求具有模块式体系结构的程序。模块式程序的一个示例是具有多个可以在运行时动态加载的模块的计帐程序。

3) 简化部署和安装。当 DLL 中的函数需要更新或修复时，部署和安装 DLL 不要求重新建立程序与该 DLL 的链接。此外，如果多个程序使用同一个 DLL，那么多个程序都将从该更新或修复中获益。当您使用定期更新或修复的第三方 DLL 时，此问题可能会更频繁地出现。

每种编程语言调用 DLL 的方法都不尽相同，在此只对用 C#调用 DLL 的方法进行介绍。首先，需要了解什么是托管，什么是非托管。一般可以认为：非托管代码主要是基于 win32 平台开发的 DLL，activeX 的组件，托管代码是基于.net 平台开发的。

因为 C#中使用 DllImport 是不能像动态 load/unload assembly 那样，所以只能借助 API 函数了。在 kernel32.dll 中，与动态库调用有关的函数包括[3]：

①LoadLibrary（或 MFC 的 AfxLoadLibrary），装载动态库。

②GetProcAddress，获取要引入的函数，将符号名或标识号转换为 DLL 内部地址。

③FreeLibrary（或 MFC 的 AfxFreeLibrary），释放动态链接库。

它们的原型分别是：

HMODULE LoadLibrary(LPCTSTR lpFileName);

FARPROC GetProcAddress(HMODULE hModule, LPCWSTR lpProcName);

BOOL FreeLibrary(HMODULE hModule);

假设有一个非托管代码的动态链接库为 Test.dll。现在，我们可以用 IntPtr hModule=LoadLibrary(“Test.dll”)；来获得 Dll 的句柄，用 IntPtr farProc=GetProcAddress(hModule,”_test@4”)；来获得函数的入口地址，注意其中“_test@4”为函数的入口地址，可以通过 Depends 工具查看。[14]

但是，知道函数的入口地址后，怎样调用这个函数呢？因为在 C#中是没有函数指针的，没有像 C++那样的函数指针调用方式来调用函数，所以我们得借助其它方法。经过研究，发现可以通过结合使用 System.Reflection.Emit 及 System.Reflection.Assembly 里的类和函数达到目的。为了以后使用方便及实现代码的复用，可以编写一个类 dld。详情见代码 dld 类。

在本系统中，HCopy.dll 和 HVite.dll 是直接从 HTK 源码中编译而成的非托管代码，在程序中，直接通过上述方式调用。

3.4.2.3 基于关键字提取的命令匹配算法

对于自然语言的大词汇量连续语音识别和控制系统而言，我们假设识别率达到了100%，也就是说假设任何话都能被正确的识别，仍然有一个问题不得不面对，这就是如何将识别结果转化成控制指令。因为识别结果是以文本形式呈现的，所以这个问题实际上是文本语言如何才能被计算机理解，继而采取相应的行动。

单纯的文本匹配是不够灵活的，因为对于人类自然语言而言，表达某个意思往往不止一种说法，比如“打开空调”就可能有以下说法：

1、开空调；

2、请打开空调；

3、请把空调打开；

4、空调被打开；

5、请你把空调打开，好吗？

6、启动空调；

7、开启空调；

8、……

当然，这里只列举了几种常见的说法，对于2、5这两种说法，实际上是有附加语，换句话说，出现了一些与句子意思关系不是很大的词，所以我们需要过滤掉；对于3、4两种说法而言，则是出现了指令的顺序颠倒，这意味着单纯的文本匹配“开空调”是不可行的；至于1、2对比就可以发现，其实两种说法都可以，更确切的说是1比2更精辟，更具有概括性，所以只要1出现了，2肯定也会出现。对于6、7两种说法，其实是对“打开空调”的另一种说法，但是仍然有效。

以上的分析说明，对于本系统而言，指令具有一定的特点，即由一个操作（Operation）和一个对象(Object)组成，比如“打开空调”，“打开”就是Operation，“空调”就是要操作的Object。因此，我们可以对文本进行关键字提取，这里的关键词就是“开”和“空调”，如果这两个词都出现了，我们就可以认为这条指令就是“开空调”。

在本系统中，具体操作如下：建立一个 CarASR_DB 数据库，里面有以下三张表 CarASR_Object、CarASR_Operation、CarASR_Command，其中前两张表为实体表，最后一张为关系表，其 E-R 图如 Fig10 所示：

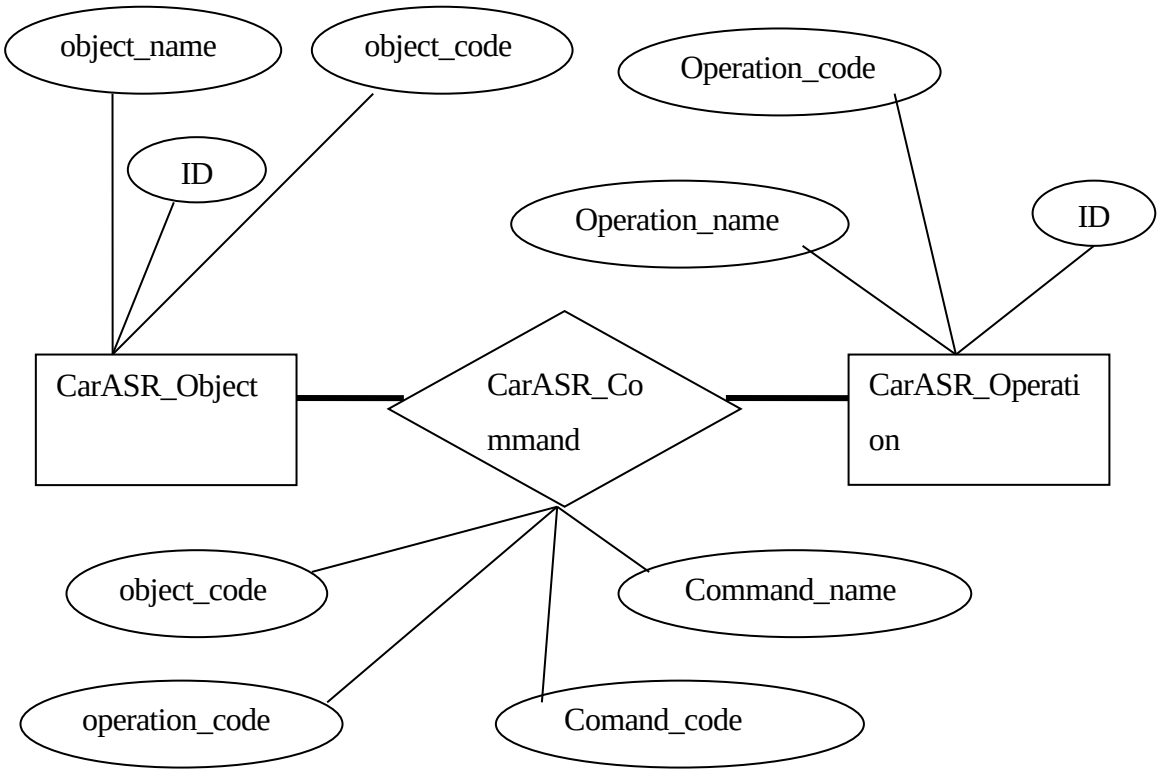

Fig10 系统数据库 E-R 图

CarASR_Object 中记录了本系统中所有可能出现的对象，比如“空调”、“雨刷”、“座椅”等等；CarASR_Operation 中则记录了本系统中所有可能出现的操作，比如“打开”、“关闭”、“报告”、“暂停”等等。CarASR_Command 中记录的是所有的操作指令，比如“开空调”、“关空调”、“开雨刷”、“关雨刷”等等。为了解决一条指令可能有多种说法的情况，比如上文中例子中的 6、7 两种说法，在 CarASR_Command 表中，“开空调”指令应该对应着多种 Operation 和 Object 的组合，即“打开空调”、“开空调”、“启动空调”、“开启空调”都对应着一条指令“开空

调”。具体实现只需要将 CarASR_Operation 表中的“打开”、“开”、“启动”、“开启”的指令代码设成一样的即可。

有了这个数据库，接下来的算法实现如 Fig11 所示：

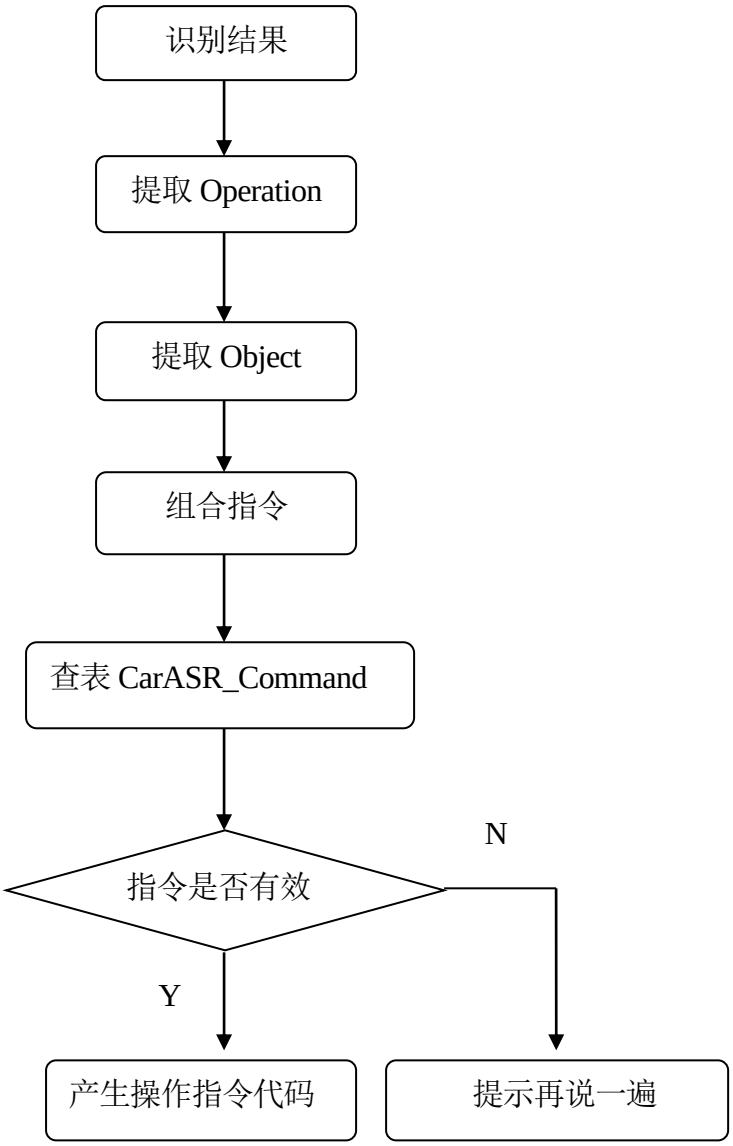

Fig11 基于关键字提取的命令匹配算法

这套基于关键字提取的命令匹配算法只是计算机理解文本语义的一种简单的方法，因为对于自然语言而言，不确定性太大，往往说法会多种，比如，假设识别结果为：

“请你把空调关上，在空调打开之后”

这样经过上面的算法，生成的指令就可能是“开空调”，这显然是不符合原意的。这时候，可能更多需要一些统计学的方法才能准确理解语义了。不过对于语音识别系统而言，对于自然语言的识别，结果能否达到100%本身也是一大难点，所以我们应该看到计算机对于语义的准确理解，也是依赖于识别结果的准确程度上的。对于本系统，毕竟是有限词汇量的连续语音识别系统，受训练数据的限制，实际上不可能会出现上面列举的多种说法的，然而，对于这种思路而言，基于关键字提取的命令匹配算法还是有意义的，所以系统选择了保留这种算法。

3.4.2.4 多线程和地下工作者（backgroundWorker）

对于Windows桌面应用程序而言，如何处理好与操作系统的关系始终是一大难点，而这一难点突出表现在进程和线程的合理使用。相比进程而言，线程显得更为灵活，在占用系统资源方面也显得更为轻量级。设计的比较好的Windows桌面应用程序，往往一条主进程下会有多条线程，主进程负责与用户交互，线程则可能在系统后台完成相应的操作，等操作完成后再通知主进程做出后续的动作。

地下工作者（backgroundWorker）是.NET架构体系中的一种特殊线程（Thread）。与一般的线程相比，它具有一个突出的特点：执行完后会触发一个事件backgroundWorker1_RunWorkerCompleted，这个事件将会由主进程来执行，这意味着在这个事件里，可以控制界面控件，而不会引发.NET里面的“跨线程操作主进程界面控件”的异常。此外，地下工作者要完成的操作可以写在方法backgroundWorker1_DoWork中完成。

对于本系统而言，多线程是必要的。因为在实时录音部分，当用户点击“实时识别”的按钮之后，出于系统交互性灵活的考虑，用户不需要再做任何界面的操作，只需要开始讲话即可。而系统至少要做出如下的操作：

1、 主界面出现“请讲话”的提示语言；

2、开始录音；

3、录音结束后主界面出现“正在识别，请等待”的提示语言；

4、开始特征提取；

5、开始语音识别；

6、提取识别结果、匹配命令、产生操作指令；

7、主界面根据返回的操作指令做出相应的界面变化，模拟控制行为。

首先，1 和 2 必须是并行的，因为录音的过程中，提示“请讲话”必须要一直存在，直到录音结束，所以可以启动一个普通进程，我们称之为录音进程，只需要执行 4 秒即可，我们假定只录 4 秒的音。然后，仔细分析会发现 3 和 4、5、6 的操作也必须是并行的，但是如果启动普通的线程来完成 4、5、6 的操作则会面临“跨线程操纵主进程控件”的危险，因为 7 必须是由主进程来完成的。如果像录音进程那样人为地设定一个普通线程的执行时长又会显得不够灵活，因为不知道识别结果会什么时候出来，也就不知道该什么时候把控制权交回给主进程。所以，这个时候就需要地下工作者了，因为它的优点刚好满足了这一需求。一方面，可以把 4、5、6 由地下工作者来完成，与主进程的 3 操作并行执行；另一方面，地下工作者执行完了之后又会触发事件，通知主进程后台的一系列识别相关操作已经完成，并且把控制权交回给主进程，完成 7 操作。这样一来，整个系统在资源利用和响应速度方面都做到了最优化。

系统的线程示例图如 Fig12 所示：

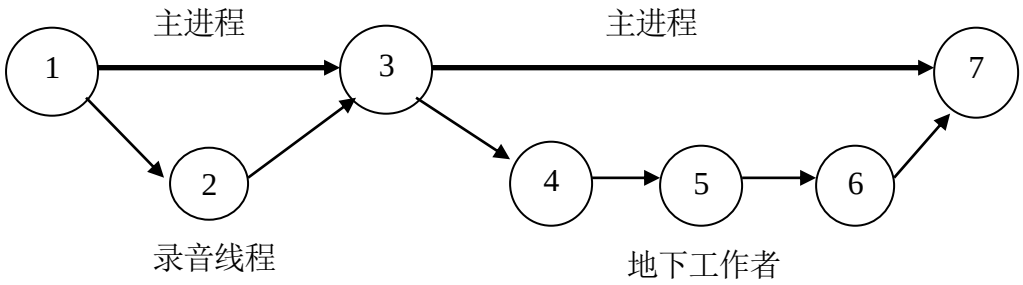

Fig12 系统进程示意图

事实证明，这种多线程的安排很合理，系统运行非常流畅。

3.4.2.5 基于 SAPI 5.0 的语音合成

SAPI SDK 是微软公司免费提供的语音应用开发工具包，这个 SDK 中包含了语音应用设计接口（SAPI）、微软的连续语音识别引擎（MCSR）以及微软的语音合成（TTS）引擎等等。目前的 5.1 版本一共可以支持 3 种语言的识别 (英语，汉语和日语)以及 2 种语言的合成（英语和汉语）。SAPI 中还包括对于低层控制和高度适

应性的直接语音管理、训练向导、事件、语法编译、资源、语音识别(SR)管理以及TTS 管理等强大的设计接口。其结构如 Fig13：

Fig13 SAPI 结构图

语音引擎则通过 DDI 层（设备驱动接口）和 SAPI(SpeechAPI)进行交互，应用程序通过 API 层和 SAPI 通信。通过使用这些 API，用户可以快速开发在语音识别或语音合成方面应用程序。SAPI5.1 SDK 可以从微软网站下载：http://www.microsoft.com/china/community/program/originalarticles/TechDoc/Cnspeech.mspx，需要安装程序的有 Speech SDK 5.1（68M）和 5.1 Language Pack（81.5M）。

SAPI5.1 的基于 Windows 平台的，通过 COM 接口进行调用。在.Net 平台下要应用 SAPI5.1，我们可以利用.Net Framework 自带的强大工具 TlbImp.exe 来把 SAPI SDK 的 COM 对象导入到.Net 中。TlbImp.exe 产生一个管制的包装类，管理客户端可以使用它。包装类管理实际的 COM 对象的参考数。当包装类当作收集的垃圾时，包装类释放掉它包装的 COM 对象。当然，你也可以在 VS.NET 环境中通过从项目参考对话框选择 COM 对象，实现 COM 对象的导入，这个过程也是通过 TlbImp.exe 来完成的。

在安装 SDK 以后，可以在 X:\Program Files\Common Files\Microsoft Shared\Speech\目录下面找到 SAPI.dll，这里面定义了 SAPI 的 COM 对象，用 Tlbimp.exe 工具将该 dll 转换成.net 平台下的 Assembly---DotNetSpeech.dll，转换的过程会提示不少的警告(warning)，但这部影响我们的开发，可以忽略。最后，我们可以用 ildasm 查看 DotnetSpeech.dll 里面的对象。[15]

得到 DotnetSpeech.dll 之后，只需添加引用到本系统项目中去即可，然后实例化一个 SpVoice 对象，将要合成语音的文本以参数传递给 SpVoice 对象的 Speak 方法即可。详情见源代码。

4 实验结果

4.1 语音识别结果评估

4.1.1 评估方案

对于连续语音而言，误识率是由插入率（Insert Error）、删除率（Delete Error）和错误率（Substitution Error）三部分组成。插入是指输入语音流在某处没有词而识别成某一个词，删除是指有词存在而没有识别出来，错误是指将某一个词识别为其他词。

语音识别中的评测一般分为正确率(Correctness)和准确率(Accuracy)：

$$CORRECTNESS = 100 \times \frac{NREF - SUB - DEL}{NREF}$$

$$ACCURACY = 100 \times \frac{NREF - SUB - DEL - INS}{NREF}$$

其中，“NREF”表示待识别词的总数，“SUB”表示替换错误的次数，“DEL”表示删除错误的次数，“INS”表示插入错误的次数。[16]

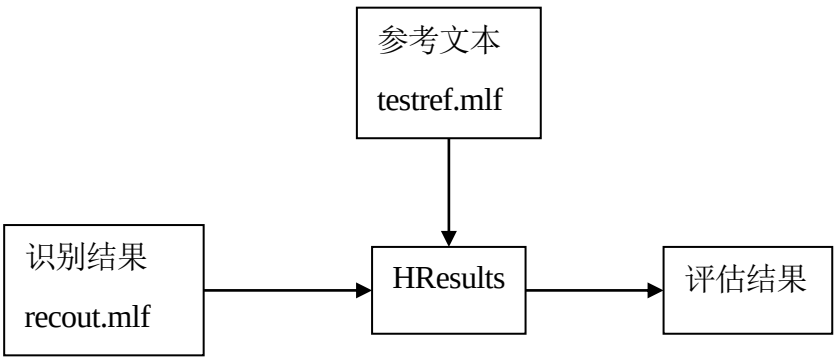

Fig14 HResults 功能示意图

HTK 中，HResults 用来评估识别结果，如 Fig14 所示，testref.mlf 文件记录的是正确的识别结果，通过 HResults 命令对比 recout.mlf 和 testref.mlf 就可以得到识别的准确率相关数据。

本系统在测试时，采用以下方案：

①声学模型

如下表所示，参与实验的声学模型如表 4 所示：

表 4 安静环境下识别实验声学模型列表

模型	强制对齐	单/三因素	GMM 数
S0	否	单音素	1
S0.align	是	单音素	1
S1.hmm44	是	三音素	4
S2.hmm44	是	三音素	4

其中 S1 模型为直接对 S0 模型进行三音素绑定之后的模型；S2 模型为利用 S1 模型对训练数据重新进行强制对齐操作，然后再进行三音素绑定后的模型； “hmmx4”表示其 HMM 状态上的 GMM（高斯混合模型）数为 x。

②测试人

CM_M 训练集内录音人，男性

GWW_F 训练集外录音人，女性

ZQS_M 训练集外录音人，男性

③测试内容：

1) 控制指令 -- 20 句
   e.g. 打开空调
   关闭雨刷
2) 温度设定 -- 10 句
   e.g. 温度设定 二十 三
   温度设定 十 五
3) 拨打电话 -- 10 句
   e.g. 拨打电话 一 三 四
4) 导航指令 -- 10 句
   e.g. 从 海上世界 到 小梅沙 怎么走

测试分两组进行，第一组为包含拨打电话指令，即一共 150 句；第二组为不包含拨打电话指令，一共是 120 句。

4.1.2 评估结果

本系统最终采用的声学模型是 S2.hmm44，即采用 4 个高斯模型的强对齐后的声学模型，实验表明：本系统不包含拨打电话指令时的识别率非常高，评估结果如下：

SENT: %Correct=95.83 [H=115, S=5, N=120]
WORD: %Corr=99.33, Acc=98.33 [H=298, D=0, S=2, I=3, N=300]

而带有拨打电话指令时，识别率会下降，评估结果如下：

SENT: %Correct=80.67 [H=121, S=29, N=150]
WORD: %Corr=97.07, Acc=85.16 [H=530, D=3, S=13, I=65, N=546]

4.1.3 结果分析

从评估结果来看，识别结果下降主要反映在拨打电话指令上，不带数字的句子识别正确率高达 95.83%，而带数字的识别率则降到了 80.67%。在实验中，我们发现，甚至只要涉及到连续的数字，识别结果必错。这是由于词网络设计过于简单造成的，导致离散的数字很难正确的组合在一起。

4.2 抗噪性能测试

4.2.1 测试方案

本测试为检验现有语音识别系统对车载噪声的抗噪性能。测试方案如下：

①模型

声学模型为原有的模型，采用干净语音（无噪声）进行训练；词网络和字典均保持不变；特征提取保持不变；其中参与测试的声学模型介绍如表 5 所示：

表 5 抗噪实验声学模型列表

模型	强制对齐	单/三因素	GMM 数
S0.align	是	单音素	1
S1.hmm124	是	三音素	12
S2.hmm44	是	三音素	4

其中， S1 模型为直接对 S0 模型进行三音素绑定之后的模型；S2 模型为利用 S1 模型对训练数据重新进行强制对齐操作，然后再进行三音素绑定后的模型；“hmmx4”表示其 HMM 状态上的 GMM（高斯混合模型）数为 x。

②测试集合

测试集合在原有的测试集合基础上叠加噪声。测试集合和 4.1 节中测试集合相同（包含拨打电话指令）；叠加噪声为汽车高速行驶（80km/h）车内的噪声；噪声与原有语音的叠加比例为 9:1（语音：噪声）。

4.2.2 测试结果

测试结果如表 6 所示：

表 6 噪声实验结果

Model	SENT (Corr)	WORD (Corr)	WORD (Acc)
S0.align	64.00	85.16	64.65
S1.hmm124	72.00	91.58	77.47
S2.hmm44	76.67	93.96	82.97
S2.hmm124	75.33	92.31	78.94

4.2.3 结果分析

从上表可以看出，本系统采用的 S2.hmm44 声学模型在添加了车载噪声后，句子识别率由 80.67%下降到了 76.67%，而词识别率由 97.07%下降到了 93.96%。虽然我们没法通过去掉 PLP 参数提取来对比识别率，但是从诸多文献中，我们还是可以看到 PLP 参数特征提取带来的抗噪效果。所以，本系统通过采用 PLP 参数特征提取的方法，是具有一定的抗噪能力的。

4.3 软件性能分析

4.3.1 系统效果图

Fig15 系统效果图

如 Fig15 所示，显示区为模拟车载指令的效果显示区域以及提示画面区域；控制区为用户交互区域，包括载入文件、手动识别和实时识别三个功能按钮；信息区则用来文本显示要识别结果以及导航信息显示区域。

4.3.2 软件性能评估

实验表明，本系统运行正常，占用系统资源较小，运行流畅。但是存在两个比较显著的问题：一是识别过程时间比较长，需要大概 6 秒；二是系统的可移植性欠佳。

经过分析，识别结果比较长是由于本系统核心的语音识别部分（特征提取、识别过程）是直接调用 HTK 中的 HCopy 和 HVite 工具编译成的动态链接库导致的，因为每次调用 HVite，系统都需要进行初始化、在内存中开辟相应的堆栈、载入声学模型、语言模型等一系列操作，这些占用了大量时间，导致大约 6 秒的等待时间；而系统的可移植

性欠佳也是由于调用了非托管代码的动态链接库，造成托管代码运行在.NET 平台上存在着一些问题。

结　论

本文基于剑桥大学的 HTK 语音工具包，成功搭建了一个有限词汇量、非特定人的中文连续语音识别系统，并以此为基础开发了一个车载环境语音识别和控制系统。并且通过学习常见的增强语音识别在噪声环境下的鲁棒性的方法，在实际系统中，为了获得一定的抗噪能力，我们采用了 PLP 参数特征向量提取的方法。

当然，本系统也存在着一定的问题，比如对于连续数字的识别，系统的识别率急剧下降，这是由于词网络过于简单造成；此外，系统识别部分响应时间过长、系统移植性欠佳等问题则是由于调用非托管代码的动态链接库所造成的。此外，对于噪声环境的分析、抗噪测试实验的完善、以及语音识别下增强噪声鲁棒性的方法等等问题将在以后的学习和进一步研究中不断改善。

参 考 文 献

[1]. 龙潜, 噪声环境下的语音识别技术研究. 2007,中国科技大学:安徽.

[2]. 王志强, 孤立词语音识别系统关键问题的研究. 2006, 北京邮电大学: 北京.

[3]. 崔文迪, 黄关维, 语音识别综述. 福建电脑, 2008(01).

[4]. 刘幺和, 宋庭新, 语音识别与控制应用技术. 2008: 科学出版社. 64~68.

[5]. Young S., K., D., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. The HTK Book. 2008; Available from: http://htk.eng.cam.ac.uk/.

[6]. 丁沛, 语音识别中的抗噪声技术. 2003, 清华大学: 北京.

[7]. J. Droppo, A. Acero, and L. Deng, Uncertainty decoding with SPLICE for noise robust speech recognition, in Proc. ICASSP. 2002, Microsoft Research: Orlando, Florida.

[8]. H. Hermansky. "Perceptual Linear Predictive (PLP) Analysis of Speech," J. Acoust. Soc. Am. 87 (4), April, 1990

[9]. M. J. F. Gales and S.J. Young, Robust continuous speech recognition using parallel model combination. IEEE Trans. on Speech and Audio Processing, 1996.

[10]. A. Acero, et al., HMM adaptation using vector Taylor series for noisy speech recognition, in Proc. ICSLP. 2000: Beijing,China.

[11]. Moreno, P.J., Speech Recognition in Noisy Environments. 1996, Carnegie Mellon University.

[12]. H. Liao and M.J.F. Gales, Issues with uncertainty decoding for noise robust speech recognition. Speech Communication, 2008(April).

[13]. 吴志勇, 蔡莲红.语音合成技术的原理. Available from: http://www.ctiforum.com/technology/tts/tts0301.htm.

[14]. Pansiom. C#程序实现动态调用 DLL 的研究. 2006; Available from: http://blog.csdn.net/pansiom/archive/2006/01/01/568096.aspx.

[15]. 陈本峰. .Net 平台下开发中文语音应用程序. Available from: http://www.microsoft.com/china/community/program/originalarticles/TechDoc/Cnspeech.mspx.

[16]. 邵阳, 王岚, 语音识别技术概述. 先进技术研究通报, 2009. **Feb**.

附录 A 外文原文

UNCERTAINTY DECODING WITH SPLICE FOR NOISE ROBUST SPEECH RECOGNITION

Jasha Droppo, Alex Acero, and Li Deng
Microsoft Research, One Microsoft Way, Redmond, Washington, USA

ABSTRACT

Speech recognition front end noise removal algorithms have, in the past, estimated clean speech features from corrupted speech features. The accuracy of the noise removal process varies from frame to frame, and from dimension to dimension in the feature stream, due in part to the instantaneous SR of the input. In this paper, we show that localized knowledge of the accuracy of the noise removal process can be directly incorporated into the Gaussian evaluation within the decoder, to produce higher recognition accuracies. To prove this concept, we modify the SPLICE algorithm to output uncertainty information, and show that combination of SPLICE with uncertainty decoding can remove 74.2% of the errors in a subset of the Aurora2 task.

1.INTRODUCTION

As soon as speech recognition systems moved out of pristine laboratory environments and into more mainstream use, it became clear that noise robustness would become a necessary component of any application. It is no longer safe to assume that speech input comes from a known microphone through a channel with high signal to noise ratio. Consequently, systems must be modified to deal with these harsher environments.

Research continues into both feature- and model-domain techniques to improve the robustness of speech recognition systems. It was shown in [1] that a feature-domain technique can achieve higher recognition accuracy than using matched noisy training and testing conditions. Since this matched condition is the limit that any model-domain technique strives for, we focus on techniques in the feature domain that allow us to beat the limit.

One general method for feature-domain cepstral de-noising is to design a module that pre-processes cepstra before they are fed into the speech recognition system. This includes

parametric feature space transformations [2, 3], spectral subtraction, vector Taylor series, CDCN, stereo piecewise linear compensation for environment (SPLICE) [4], and cepstral smoothing techniques such as RASTA and CMN. The advantage of all of these techniques is that they can be seamlessly integrated into existing systems, without a complete overhaul of existing code.

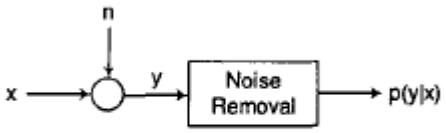

Fig. 1. Generic Noise Removal Framework

Many of these algorithms can be simply modified to produce estimates of the uncertainty of the noise removal process in addition to the cleaned features. Figure 1 represents the proposed system. Noise n corrupts the clean speech signal x, producing the noisy signal y. This is followed by the noise removal algorithm, which outputs a conditional PDF P(y|x).

This paper describes how the uncertainty from the noise removal system can be directly integrated into the decoding process. The uncertainty is generated by augmenting the SPLICE algorithm, which recently had the best noise removal performance at a special Aurora session in Eurospeech 2001 [4]. Uncertainty decoding improves the performance even further.

Our work generalizes the missing feature techniques in [5],and integrates more easily into existing decoders. It has a continuous, soft weighting of data rather than discarding regions below a set threshold.

In Section 2, we describe how the Gaussian evaluation in the decoder is modified to incorporate the new information from the front end. Section 3 contains a brief overview of SPLICE and a description of how the uncertainty estimation can be integrated into the noise removal process. In Section 4, we provide experimental results that show how uncertainty decoding improves recognition accuracy in the Aurora task.

2 .UNCERTAINTY DECORDING

Here we present a framework whereby estimates of the error, or uncertainty associated with noise removal can be incorporated into the recognition process.

2.1 Uncertainty Modifies Gaussian Evaluation

At the heart of the speech recognition engine, many Gaussian mixture components are evaluated.

When recognizing uncorrupted speech cepstra, the purpose of these evaluations is to discover the probability of each clean observation vector, conditioned on the mixture index, $p_{x|m}(x|m)$ for the individual Gaussian in the speech model used by the recognizer.

If the training and testing conditions do not match, as is the case in noise-corrupted speech recognition, one option is to ignore the imperfections of the noise removal, and evaluate $p_{x|m}(\hat{x} \mid m)$. This is the classic case of passing the output of the noise removal algorithm directly to the recognizer.

A more rigorous approach, uncertainty decoding, is to generate the joint conditional PDF p(x,y|m) and marginalize over all possible unseen clean speech cepstra:

$$p(y \mid m) = \int_{-\infty}^{\infty} p(y, x \mid m) dx$$

Under this framework, instead of just providing cleaned cepstra, the noise removal process estimates the conditional distribution p(y|x, m), as a function of x. For ease of implementation, we assume that p(y|x) is independent of m:

$$p(y \mid x, m) \approx p(y \mid x) = \alpha N(x; \hat{x}, \delta_{\hat{x}}^2)$$

Finally, the probability for the observation y, conditioned on each acoustic model Gaussian mixture component m, can be calculated.

$$\begin{aligned} p(\mathbf{y}|m) &= \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{x}, m) p(\mathbf{x}|m)\, d\mathbf{x} \\ &= \alpha \int_{-\infty}^{\infty} N(\hat{\mathbf{x}}; \mathbf{x}, \sigma_{\hat{\mathbf{x}}}^2) N(\mathbf{x}; \mu_m, \sigma_m^2)\, d\mathbf{x} \\ &= \alpha N(\hat{\mathbf{x}}; \mu_m, \sigma_m^2 + \sigma_{\hat{\mathbf{x}}}^2) \qquad (1) \end{aligned}$$

This formula is evaluated for each Gaussian mixture component in the decoder, $p(x \mid m) = N(x; \mu_m, \delta_m^2)$.

The uncertainty output from the front end increases the variance of the Gaussian mixture component, producing an effective smoothing in cases where the front end is uncertain of the true value of the cleaned cepstra.

2.2 Special Cases

Two special cases exist for uncertainty decoding. In the absence of uncertainty information from the noise removal process, we can either assume that there is no uncertainty or that there is complete uncertainty.

If there were no uncertainty, then $\delta^2_{\hat{x}} = 0$. The probability of the observation y, for each acoustic model Gaussian mixture component m, simplifies to:

$$p(\mathbf{y}|m) = p(\hat{\mathbf{x}}|m) = N(\hat{\mathbf{x}}; \mu_m, \sigma_m^2). \qquad (2)$$

This is the traditional method of passing features directly from the noise removal algorithm to the decoder.

If there were complete uncertainty of any of the censtral coefficients, the corresponding $\delta^2_{\hat{x}}$ would approach infinity. That coefficient would have no effect on the calculation of p(y|m). This is desirable behavior, under the assumption that the coefficient could not contribute to discrimination.

Both of these extreme cases are similar to the computations performed when using hard thresholds with missing feature techniques [5]. There has been some success in incorporating heuristic soft thresholds with missing feature techniques[6], but we believe that uncertainty decoding benefits from a rigorous probabilistic framework.

3.SPLICE NOISE REMOVAL AND UNCERTAINTY

SPLICE [4] is an algorithm that learns a probabilistic model of distortion from a clean cepstral vector, x, into a noisy one, y. Using this model, SPLICE can produce an approximation of the PDF p(y|x) for any distorted input y.

3.1 A Model of Speech and its Degradation

SPLICE makes two fundamental assumptions about the form of the joint probability of x and y. The first assumption is that the noisy speech cepstral vector follows the distribution of mixture of Gaussians:

$$p(y) = \sum_s N(y; \mu_s, \sigma_s)\, p(s)$$

One distribution p(y) is trained for each separate distortion condition (not indexed for clarity), and can be thought as a "codebook" with a total of N codewords (means) and their variances. Each codebook is implicitly conditioned on a specific noise type and level. To select the appropriate codebook at runtime, we developed an effective on-line environmental selection method, which has been described in detail in [7].

The second assumption made by the SPLICE is that the conditional probability density function (PDF) for the clean vector x given the noisy speech vector, y, and the region index, s, is Gaussian whose mean vector is a linear transformation of the noisy speech vector y. In this paper, we take a simplified form of this linear transformation by making the rotation matrix to be the identity matrix, leaving only the bias or correction vector .Thus, the conditional PDF is assumed to have the form.

$$p(x \mid y, s) = N(x; y + r_s, \Gamma_s^2) \qquad (3)$$

3.2 SPILCE Training

Since the noisy speech PDF p(y) is assumed to be a mixture of Gaussians, the standard EM algorithm can be used to train μ_s and σ_s on noisy speech. Initial values of the parameters are determined by a VQ clustering algorithm.

If stereo data is available, the parameters r_s and Γ_s of the conditional PDF p(x|y, s) can be trained using the maximum likelihood criterion:

$$\mathbf{r}_s = \frac{\sum_n p(s|\mathbf{y}_n)(\mathbf{x}_n - \mathbf{y}_n)}{\sum_n p(s|\mathbf{y}_n)} \qquad (4)$$

$$\mathbf{\Gamma}_s = \frac{\sum_n p(s|\mathbf{y}_n)(\mathbf{x}_n - \mathbf{y}_n)^2}{\sum_n p(s|\mathbf{y}_n)} - \mathbf{r}_s^2 \qquad (5)$$

$$p(s|\mathbf{y}_n) = \frac{p(\mathbf{y}_n|s)p(s)}{\sum_s p(\mathbf{y}_n|s)p(s)} \qquad (6)$$

This training procedure requires a sat of stereo (two channel) data. One channel contains the clean utterance, and the other contains the same utterance with distortion. The two-channel data can be collected, for example, by simultaneously recording on one close-talk and one far-field microphone.

For the Aurora work reported in this paper, the SPLICE parameters were trained using identical utterances from the clean training set and the multi-style training set.

3.3 Complete SPLICE

In the past, SPLICE has been applied to the 13-dimensional static cepstral coefficients only, ignoring the fact that delta and acceleration coefficients are also central to the recognition process. Since SPLICE processes each frame independently, one could argue that the SPLICE mapping is incomplete. The delta and acceleration features computed on-line during recognition, from these cleaned static features, are not optimal.

Alternatively, the static feature vector can be completed with delta and acceleration components before passing the vector to SPLICE for processing. Under this improved scenario, SPLICE maps a 39-dimensional noisy input vector to a 39-dimensional cleaned speech output vector. The advantage of this approach is that it is consistent across the entire vector being modeled by the recognizer. Of course, the delta and acceleration parameters no longer correspond to a linear filtering of the static parameters.

3.4 Estimating p(y|x)

For uncertainty decoding, the front end must provide an estimate of the conditional probability density function

$$p(\mathbf{y}|\mathbf{x}) = \frac{\sum_s p(\mathbf{x}|\mathbf{y},s)p(\mathbf{y}|s)p(s)}{p(\mathbf{x})}.$$

as a function of x. We do this by leveraging the probabilistic framework of SPLICE.

Each term of the numerator is directly computable from the SPLICE parameters. 1t is somewhat more difficult to derive the prior p(x).

First, note that the joint conditional probability p(x,y|s) can be re-written as,

$$\begin{aligned} p(\mathbf{x},\mathbf{y}|s) &= p(\mathbf{x}|\mathbf{y},s)p(\mathbf{y}|s) \\ &= N(\mathbf{x};\mathbf{y}+\mathbf{r}_s,\Gamma_s^2)N(\mathbf{y};\mu_s,\sigma_s^2) \\ &= N(\mathbf{y};\frac{\sigma_s^2(\mathbf{x}-\mathbf{r}_s)+\Gamma_s^2\mu_s}{\Gamma_s^2+\sigma_s^2},\frac{\Gamma_s^2\sigma_s^2}{\Gamma_s^2+\sigma_s^2}) \\ & \quad N(\mathbf{x};\mu_s+\mathbf{r}_s,\Gamma_s^2+\sigma_s^2) \end{aligned}$$

So the prior for x is simply,

$$
\begin{aligned}
p(\mathbf{x}) &= \sum_s p(\mathbf{x}|s)p(s) \\
&= \sum_s \int_{-\infty}^{\infty} p(\mathbf{x},\mathbf{y}|s)p(s)\,d\mathbf{y} \\
&= \sum_s N(\mathbf{x};\mu_s+\mathbf{r}_s,\Gamma_s^2+\sigma_s^2)p(s)
\end{aligned}
$$

In order to simplify p(y|x) for use in Eq. 1, we approximate this mixture of Gaussians as a single Gaussian,

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{x}) &= \frac{\sum_s p(\mathbf{x}|\mathbf{y},s)p(\mathbf{y}|s)p(s)}{p(\mathbf{x})} \\
&= \frac{\sum_s N(\mathbf{x};\mathbf{y}+\mathbf{r}_s,\Gamma_s^2)p(\mathbf{y}|s)p(s)}{N(\mathbf{x};\mu_\mathbf{x},\sigma_\mathbf{x}^2)} \\
&= \sum_s N(\mathbf{x};\hat{\mathbf{x}}_s,\sigma_{\hat{\mathbf{x}}_s}^2)p(\mathbf{y}|s)p(s), \text{ where} \qquad (7) \\
\hat{\mathbf{x}}_s &= \frac{\sigma_\mathbf{x}^2(\mathbf{y}+\mathbf{r}_s)-\Gamma_s^2\mu_\mathbf{x}}{\sigma_\mathbf{x}^2-\Gamma_s^2}, \text{ and} \qquad (8) \\
\sigma_{\hat{\mathbf{x}}_s}^2 &= \frac{\sigma_\mathbf{x}^2\Gamma_s^2}{\sigma_\mathbf{x}^2-\Gamma_s^2}. \qquad (9)
\end{aligned}
$$

Of course this derivation only makes sense when

$$\delta_X^2 \geq \Gamma_s^2 \qquad (10)$$

Recall that $\sigma_\mathbf{x}^2$ is the global variance of the clean speech prior, and that Γ_s^2 is the expected squared error of the noise removal process. There are two cases where we might expect Eq. l0 to be violated. Either the noise removal process is fundamentally flawed, and expects itself to be doing worse than outputting the prior mean for speech, or our approximation of p(x) is causing mischief. In the handful cases that don't obey Eq. 10, we assume the latter to be the case, and simply force $\sigma_\mathbf{x}^2 \geq \Gamma_s^2 + \varepsilon$, where $\varepsilon = 0.1$. In practice, this occurs on less than 5% of cepstral coefficients.

Ideally, the conditional distribution we seek would be given by the sum in Eq. 7, Gut the fast implementation we use the assumption that p(y|s)p(s) is zero for all but one value of s.

$$\hat{p}(\mathbf{y}|s)p(s) \cong \begin{cases} 1 & s = \text{argmax}_s p(\mathbf{y}|s)p(s) \\ 0 & \text{otherwise} \end{cases}$$

SPLICE processing then consists of two sequential operations. First. finding optimal codeword s using the VQ codebook based on the parameters (μ_s, σ_s), and then finding $\hat{\mathbf{x}}$ and $\delta_{\hat{\mathbf{x}}}^2$ according to Eq.8 and Eq. 9.

4. Result

4.1 Qualitative

Figure 2 illustrates that SPLICE is producing reasonable outputs. The upper half of the figure shows c_0 as a function of time, for both uncorrupted and corrupted speech. The lower half of the figure shows the $\hat{\mathbf{x}}$ parameter output by SPLICE, as well as the range of variation($\pm\sigma_x^2$).

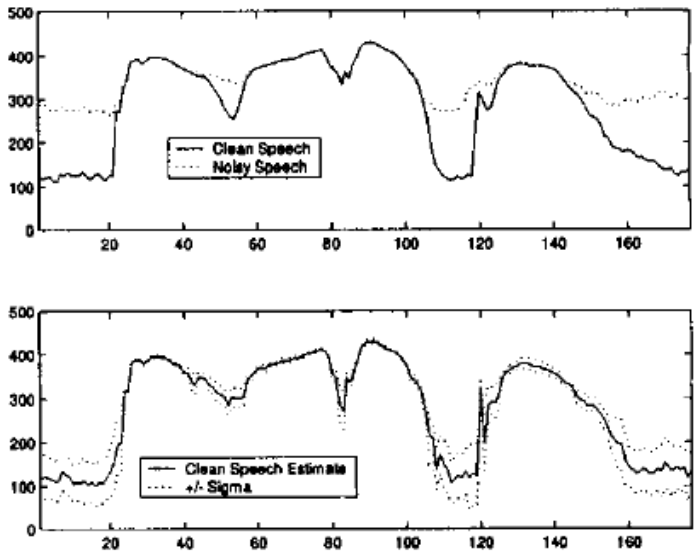

Fig. 2. C_0 for clean speech (top), together with $p(y|x)$ for a corresponding noisy utterance, plotted as a function of x.

The cepstral coefficient c_0 is roughly related to frame energy. In speech regions, c_0 is large and masks the contribution of noise. The value of $\hat{\mathbf{x}}$ is accurate, and the margin of error is small. In non-speech regions, c_0 from the noise masks the speech value. The value of $\hat{\mathbf{x}}$ is less

accurate, and is accompanied by a larger variance. We observe that the c_0 for clean speech is consistent with the bounds described by the dashed lines in the lower half of Figure 2.

4.2 Quantitative

Several connected digit experiments were run using the framework provided in the Aurora2[8] corpus. The acoustic model training data consists of 8440 clean utterances and the same utterances in groups of 422, corrupted 20 different ways. These 20 sets consist of four noise types (subway, babble, car, and exhibition) artificially added at five different signal to noise ratios (infinite, 20, 15, 10, and 5). All of our tests were performed with an acoustic model trained on clean data with scripts provided with the Aurora2 framework.

Only results on set A, which consists of 28028 files are reported in this paper. This set is partitioned in to the same noise types found in the training data, artificially added at seven different signal to noise ratios (infinite, 20, 15, l0, 5, 0, and -5).

SPLICE, as represented in this paper, does not generalize well to unseen noise types (contained in set B), or unseen convolutional channels (contained in set C), although it has been shown that both of these limitations can be easily overcome [4] with noise mean normalization (NMN) and cepstral mean normalization (CMN). Neither technique is used in this paper; in theory the results on set. A should be unchanged by their omission.

Table 1 shows digit recognition accuracy for eight different experiments, consisting of three different front ends, and three different decoding strategies.

Table 1. A comparison of the digit accuracy of eight front end configurations on Aurora test set A, with a clean acoustic model.

	Uncertainty Source		
Front End	None	SPLICE	True
Standard MFCC	63.66%	N/A	85.94%
SPLICE	87.22%	87.47%	89.52%
Complete SPLICE	88.21%	90.63%	92.81%

The front ends considered were a standard MFCC algorithm, SPLICE, and Complete SPLICE. The standard MFCC algorithm was identical to the Aurora reference in all respects but one: we modified it to use magnitude-squared spectra internally instead of magnitude spectra. SPLICE and Complete SPLICE post-process this data as described in Section 3.

We considered decoders without uncertainty, with uncertainty generated by SPLICE, and true uncertainty generated from oracle data. The SPLICE uncertainties were of course unavailable for the reference MFCC front end, and computed as described above for the other front ends. The true uncertainties were derived by computing the magnitude-squared error between the front-end output and the true clean speech cepstra, which are available in the Aurora2 data.

The experiments with true uncertainty are indicative of how much improvement we can expect by performing uncertainty decoding. For the standard front end, it is possible to eliminate over 60% of the word error rate just by adding perfect knowledge of the magnitude of the cepstral errors. For the front ends containing SPLICE, the possible improvement is smaller, but not negligible.

Estimating the uncertainty using SPLICE yields improvement in both cases. The best realizable system, the Complete SPLICE front end, including uncertainty decoding, reduces the word error rate by 74.2% with respect to the front end without SPLICE.

5.DISCUSSION

We have described a systematic process for incorporating uncertainty from the noise removal process into a speech recognition system.

Uncertainty decoding carries two major benefits. First, including the conditional probability p(y|x) effectively accounts for the residual corruption from the noise removal process. And second, because uncertainty decoding is Eased on a comprehensive probabilistic framework, we avoid any heuristic tuning.

It should be simple to modify most noise removal algorithms to produce uncertainty estimates. We have demonstrated this using SPLICE. Introducing uncertainty decoding reduced the average word error rate of our state of the art system by over 20% relative.

6.REFERENCE

[1]L.Deng,A.Acero,M.Plumpe,and X.D.Huang,"Large vocabulary speech recognition under adverse acoustic environments," in Proc.2000 ICSLP,Beijing,China,October 2000,pp.806-809.

[2]Y.Ephraim and M.Rahim,"On second-order statistics and linear estimation of cepstral coefficients," IEEE Trans.Speech and Audio Proc.,vol.7,no.2,pp.162-176,March 1999.

[3]Y.Zhao,"Frequency-domain maximum likelihood estimation for automatic speech recognition in additive and convolutive noises,"IEEE Trans. Speech and Audio Proc.,vol.8,no.3,pp.255-266,May 2000.

[4]J.Droppo,L.Deng,and A.Acero,"Evaluation of the SPLICE algorithm on the Aurora2 database(web update),"in Proc.Eurospeech 2001,Aalborg,Denmark,September 2001,pp.217-220.

[5]M.Cooke,P.Green,L.Josifovski,and A.Vizinho,"Robust automatic speech recognition with missing and unreliable acoustic data,"Speech Communication,vol.34,no.3,pp.267-285,June 2001.

[6]P.Green J.P.Barker,M.Cooke,"Robust ASR based on clean speech models:An evaluation of missing data techniques for connected digit recognition in noise,"in Proc.Eurospeech 2001,Aalborg,Denmark,September 2001,pp.213-216.

[7]J.Droppo,A.Acero,and L.Deng,"Efficient on-line acoustic environment estimation for FCDCN in a continuous speech recognition system," in Int.Conf.On Acoustics,Speech and Signal Processing,Salt Lake City,Utah,May 2001.

[8]H.G.Hirsch and D.Pearce,"The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions," in ISCA ITRW ASR2000"Automatic Speech Recognition:Challenges for the Next Millennium",Paris,France,September 2000.

附录 B 外文译文

基于 SPLICE 算法的非确定性声学解码的噪声环境下语音识别的鲁棒性研究

Jasha Droppo,Alex Acero,and Li Deng
微软研究院，美国华盛顿 RedMond 微软研发总部

摘要

在过去，语音识别的噪声去除算法已经可以从受干扰的语音特征中提取干净的语音特征信息。噪声去除过程的精度却取决于即时识别过程中的每一帧、每一维的特征流。在这篇文章中，我们展示了把噪声去除算法直接整合到解码器中的高斯公式中的基础知识，以此来产生更高的识别精度。为了证明这一概念，我们修改了 SPLICE 算法来输出不确定信息，并且显示把 SPLICE 算法和非确定性声学解码组合在一起可以降低 74.2%的识别错误，我们的实验环境是 Aurora2 项目中的一个子集。

1、介绍

一旦语音识别系统从原始的实验室环境移植到更多主流的应用当中，很明显，对于噪声的鲁棒性将成为任何应用的一个必要的组成部分。我们对于语音输入是来自一个已知的具有高信噪比的麦克风的假设不再乐观。结果，系统必须要加以修改以适应更粗糙的环境。

我们的研究继续基于特征域和模型域两方面的技巧，以此来提高语音识别系统的鲁棒性。从[1]中我们可以看到，一种特征域的技巧比起使用匹配的噪声训练和测试条件而言，可以达到更高的识别精度。既然这种匹配条件是任何基于模型域方法所努力追求的极限，我们将通过关注特征域的技巧以此来突破这种极限。

对于特征域的倒频谱的去噪思路而言，一种通常的做法是设计一个在信号被送往语音识别系统之前的预处理倒频谱的模块。这就包括参数化的特征空间转换[2,3]，光谱差减法，泰勒级数展开的近似，CDCN，立体化的分段环境线性补偿技术（SPLICE）

[4]，此外，增强的倒频谱技术还有比如 RASTA 和 CMN。这些技巧的优势便是它们可以无缝的整合到现有的系统中，不需要完全修改现有代码。

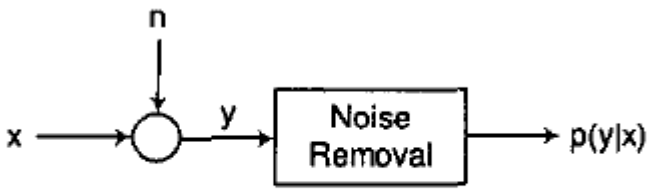

Fig. 1. Generic Noise Removal Framework

这些算法中许多经过简单的修改之后，用来估计除去干净特征之外的噪声去除算法的不确定性。图 1 就代表了这种假设的系统。噪声 n 侵蚀了干净的语音信号 x，产生了包含噪声的语音信号 y。后面紧跟着的是噪声去除算法，输出的是 p(y|x)的概率密度函数。

这篇文章描述了如何把噪声去除算法产生的不确定性直接整合到解码过程中。这种不确定性是通过修改 SPLICE 算法产生的，这种算法进来在 2001 年欧洲语音期刊中的 Aurora 章节中显示出最好的去噪表现。而非确定性声学解码更提高了这种良好的表现。

我们的工作也产生基于遗失特征的一些提取技巧，这些技巧可以在[5]中查阅，同时我们的技巧可以很方便的整合到现有的解码器中。这种做法比起设定一个门限来产生丢弃区域而言有着连续的、轻量级的数据量。

在第二部分，我们将阐述解码器中的高斯公式是怎样被修改以使前端处理包含新的信息。第三部分包含了一个简短的介绍 SPLICE 算法以及描述了怎样把不确定性估计整合到去噪过程中。第四部分，我们提供了实验数据来显示非确定性声学解码在 Aurora 项目中提高了识别精度。

2、非确定性声学解码

在这里我们展示了一个借助对错误率的估计，或者说不确定性，连同把去噪算法整合到识别过程中的框架。

2.1 不确定性估计对高斯公式的改进

在语音识别引擎的核心，许多高斯混合模块都被改进了。

当识别没有被损坏的语音倒频谱信息时，这些改进的目的是发现每一个观察向量的概率，这一概率取决于混合索引，也就是被识别器所使用的语言模型中的单独高斯函数 $p_{x|m}(x|m)$。

如果训练条件和测试条件不匹配，就像在噪声干扰的语音识别中情况一样，一个选择是忽略噪声去除过程的缺点，估计 $p_{x|m}(\hat{x}\,|\,m)$。这是把噪声去除算法的输出直接送到识别器的经典情况。

一种更严厉的方式，非确定性声学解码，是产生联合概率密度 p(x,y|m)，并且排斥所有可能的隐藏的语音倒频谱：

$$p(y|m) = \int_{-\infty}^{\infty} p(y, x|m)dx$$

在这个框架之下，与其单纯提供干净倒频谱信息，不如去除噪声过程直接估计条件概率 p(y|x,m)，这是关于 x 的一个函数。为了实现的简便，我们假设 p(y|x)独立于 m：

$$p(y|x,m) \approx p(y|x) = \alpha N(x;\hat{x},\sigma_{\hat{x}}^2)$$

最后，观察向量 y，取决于每一个声学模型的高斯混合模型 m，就可以被计算出来了。

$$\begin{aligned} p(\mathbf{y}|m) &= \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{x},m)p(\mathbf{x}|m)\,d\mathbf{x} \\ &= \alpha\int_{-\infty}^{\infty} N(\hat{\mathbf{x}};\mathbf{x},\sigma_{\hat{\mathbf{x}}}^2)N(\mathbf{x};\mu_m,\sigma_m^2)\,d\mathbf{x} \\ &= \alpha N(\hat{\mathbf{x}};\mu_m,\sigma_m^2+\sigma_{\hat{\mathbf{x}}}^2) \qquad (1) \end{aligned}$$

这个公式是基于每一个解码器中高斯混合模块来进行评价 $p(x\,|\,m) = N(x;\mu_m,\delta_m^2)$。

这种前端的不确定性估计的输出增加了高斯混合模块的方差，当前端对真实的干净倒频谱信息不确定时，会对结果产生一个有效的加强。

2.2 特殊情况

两种特殊情况存在于非确定性声学解码。在噪声处理过程缺失不确定性的信息的情况时，我们可以认为要么不存在着不确定性，要么完全是不确定性。

如果完全没有不确定性，那么 $\delta^2_{\hat{x}} = 0$。观察向量 y 的概率，对每一个声学模型高斯混合模块 m，可以简化为：

$$p(\mathbf{y}|m) = p(\hat{\mathbf{x}}|m) = N(\hat{\mathbf{x}}; \mu_m, \sigma_m^2). \qquad (2)$$

这是传统的直接把特征向量从噪声去除算法送往解码器的做法。

如果对于任何的倒频谱系数都存在着不确定性，相应的$\delta^2_{\hat{x}}$将会接近无穷大。那样系数将会对 p(y|m)的计算没有任何影响。这是期望的行为，因为假设就是系数对判别没有任何作用。

以上两种极端情况都和使用设定门限值、运用隐藏特征向量的技巧类似，这在[5]中有详细介绍。现在，已经成功研究出了结合启发式软门限、运用隐藏特征向量的方法[6]，但是我们仍然相信非确定性声学解码是对严格的概率框架有好处的。

3、SPLICE 噪声去除算法和非确定性声学解码

SPLICE[4]是一种算法，它可以得到从干净的倒频谱向量 x，扭曲到带噪声的倒频谱向量 y 过程中的概率模型。使用这个概率模型，SPLICE 就可以对任何扭曲的输入向量 y 产生概率密度函数 p(y|x)。

3.1 一个语言模型及其简化

SPLICE 对 x 和 y 的联合概率做了两个基本的假设。第一个假设是噪声语言频谱向量遵循混合高斯分布，即：

$$p(y) = \sum_s N(y; \mu_s, \sigma_s)\, p(s)$$

一个 p(y)的分布被用来在分离的扭曲条件下进行训练（不是声明过的索引），并且被想象成一个“电报密码本”，一共有 N 个密语（平均值）和方差。每个“电报密码本”都隐藏地取决于一种特殊的噪声类型和级别。为了实时的选择合适的“电报密码本”，我们研究了一种有效地在线环境选择方法，这在[7]中有详细介绍。

第二个假设是在给定噪声语音向量 y 以及区域索引 s 下，得到干净倒频谱信息向量 x 的条件概率密度函数(PDF)，也是满足高斯分布的，并且其均值向量是噪声语音向量 y

的一个线性转换。在这篇文章中，我们简单的认为这种线性转换是把旋转矩阵转化为单位矩阵，只留下偏差和错误向量。因此，条件概率密度函数被认为具有以下形式：

$$p(x \mid y, s) = N(x; y + r_s, \Gamma_s^2) \qquad (3)$$

3.2 SPLICE 训练

既然噪声语音下的概率密度函数 p(y)被假定是一个高斯混合函数，标准的 EM 算法可以用来训练噪声环境下的μ_s和σ_s。参数的初值可以通过 VQ 聚类算法决定。

如果立体声数据可以得到，条件概率密度函数 p(x|y,s)的参数r_s 和Γ_s可以通过最大似然判别法训练得到：

$$\mathbf{r}_s = \frac{\sum_n p(s|\mathbf{y}_n)(\mathbf{x}_n - \mathbf{y}_n)}{\sum_n p(s|\mathbf{y}_n)} \qquad (4)$$

$$\mathbf{\Gamma}_s = \frac{\sum_n p(s|\mathbf{y}_n)(\mathbf{x}_n - \mathbf{y}_n)^2}{\sum_n p(s|\mathbf{y}_n)} - \mathbf{r}_s^2 \qquad (5)$$

$$p(s|\mathbf{y}_n) = \frac{p(\mathbf{y}_n|s)p(s)}{\sum_s p(\mathbf{y}_n|s)p(s)} \qquad (6)$$

训练过程需要一系列的立体声（双信道）的数据。一个信道包含干净语音，另一个包含扭曲后的该语音。这种两信道的数据可以通过在近距离和远距离两种情况放置麦克风即时讲话录制得到。

对于这篇文章中提到的 Aurora 工作，SPLICE 参数可以通过在干净训练集中和多形式训练集中选取相同的语料来进行训练得到。

3.3 完全 SPLICE

在过去，SPLICE 只被应用于 13 维的静态倒频谱系数，而忽略了变化量和加速量系数同样对识别过程至关重要的事实。既然 SPLICE 是单独的处理每一帧的，我们可以认为 SPLICE 的映射是不完全的。那么，那些从干净静态特征中得到的实时识别中的变化量和加速量，也就不是可有可无的了。

既然这样，静态特征向量就应该先经过计算变化量和加速量的模块，再被送往 SPLICE 进行处理。在这种改进的情况下，SPLICE 映射一个 39 维噪声输入向量到一个

39 维的干净语音输出向量。这种方式的优点是在被识别器建模时它是连续的经过整个向量的。当然，变化量和加速量不再对应于一个静态参数的线性过滤。

3.4 估计 p(y|x)

对于非确定性声学解码，前端必须要提供一个条件概率密度函数的估计：

$$p(\mathbf{y}|\mathbf{x}) = \frac{\sum_s p(\mathbf{x}|\mathbf{y},s)p(\mathbf{y}|s)p(s)}{p(\mathbf{x})}.$$

它是 x 的一个函数。我们通过作用于 SPLICE 的条件框架来得到这一估计。

分子的每一项都直接通过 SPLICE 的参数计算得到。某种程度上，先验概率 p(x)是更难得到的。

首先，注意联合条件概率 p(x,y|s)可以被重写为：

$$\begin{aligned} p(\mathbf{x},\mathbf{y}|s) &= p(\mathbf{x}|\mathbf{y},s)p(\mathbf{y}|s) \\ &= N(\mathbf{x};\mathbf{y}+\mathbf{r}_s,\Gamma_s^2)N(\mathbf{y};\mu_s,\sigma_s^2) \\ &= N(\mathbf{y};\frac{\sigma_s^2(\mathbf{x}-\mathbf{r}_s)+\Gamma_s^2\mu_s}{\Gamma_s^2+\sigma_s^2},\frac{\Gamma_s^2\sigma_s^2}{\Gamma_s^2+\sigma_s^2}) \\ & \quad N(\mathbf{x};\mu_s+\mathbf{r}_s,\Gamma_s^2+\sigma_s^2) \end{aligned}$$

因此 x 的先验概率就简单了，

$$\begin{aligned} p(\mathbf{x}) &= \sum_s p(\mathbf{x}|s)p(s) \\ &= \sum_s \int_{-\infty}^{\infty} p(\mathbf{x},\mathbf{y}|s)p(s)\,d\mathbf{y} \\ &= \sum_s N(\mathbf{x};\mu_s+\mathbf{r}_s,\Gamma_s^2+\sigma_s^2)p(s) \end{aligned}$$

为了简化 p(y|x)以便在等式 1 中使用，我们估计这个混合高斯函数为一个单独的高斯函数，

$$\begin{aligned} p(\mathbf{x}) &\approx N(\mathbf{x};\mu_\mathbf{x},\sigma_\mathbf{x}^2), \text{where} \\ \mu_\mathbf{x} &= \sum_s (\mathbf{r}_s+\mu_s)\,p(s), \text{and} \\ \sigma_x^2 &= \sum_s \left((\mathbf{r}_s+\mu_s)^2+\sigma_s^2+\Gamma_s^2\right)p(s)-\mu_\mathbf{x}^2. \end{aligned}$$

然后，我们使用这个估计来简化 p(y|x)。

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{x}) &= \frac{\sum_s p(\mathbf{x}|\mathbf{y},s)p(\mathbf{y}|s)p(s)}{p(\mathbf{x})} \\
&= \frac{\sum_s N(\mathbf{x};\mathbf{y}+\mathbf{r}_s,\Gamma_s^2)p(\mathbf{y}|s)p(s)}{N(\mathbf{x};\mu_\mathbf{x},\sigma_\mathbf{x}^2)} \\
&= \sum_s N(\mathbf{x};\hat{\mathbf{x}}_s,\sigma_{\hat{\mathbf{x}}_s}^2)p(\mathbf{y}|s)p(s), \text{ where} \qquad (7) \\
\hat{\mathbf{x}}_s &= \frac{\sigma_\mathbf{x}^2(\mathbf{y}+\mathbf{r}_s)-\Gamma_s^2\mu_\mathbf{x}}{\sigma_\mathbf{x}^2-\Gamma_s^2}, \text{ and} \qquad (8) \\
\sigma_{\hat{\mathbf{x}}_s}^2 &= \frac{\sigma_\mathbf{x}^2\Gamma_s^2}{\sigma_\mathbf{x}^2-\Gamma_s^2}. \qquad (9)
\end{aligned}
$$

当然，这个推导成立的的条件是：

$$\delta_X^2 \geq \Gamma_s^2 \qquad (10)$$

这里σ_x^2是干净语音的全局方差，Γ_s^2是噪声去除过程中的期望的错误的平方。有两种情况我们可以认为等式 10 不成立。要么是认为噪声去除过程有缺陷，并且认为它本身比输出先验语音均值表现更差；要么使我们对 p(x)的估计引起了错误。在大量的违背等式 10 的案例中，我们假设是后者这种情况发生，并且简单的要求$\sigma_x^2 \geq \Gamma_s^2 + \varepsilon$，其中$\varepsilon = 0.1$。在实际中，这在少于 5%的倒频谱系数中发生。

理想情况，我们所追求的条件分布将会通过等式 7 得到，但是，我们对于 p(y|s)p(s)的假设只对于 s 的一种情况不等于 0。

$$\hat{p}(\mathbf{y}|s)p(s) \cong \begin{cases} 1 & s = \operatorname{argmax}_s p(\mathbf{y}|s)p(s) \\ 0 & \text{otherwise} \end{cases}$$

SPLICE 的后续处理由两个序列操作构成：一是找到通过使用 VQ 密码本找到可选的密语 s，这是基于参数(μ_s, σ_s)的；二是通过等式 8 和 9 找到$\hat{x}$和$\delta_{\hat{x}}^2$。

4、结果

4.1 定性分析

图 2 显示 SPLICE 产生了合理的输出。图的上半部分显示了c_0是时间的函数，对干净语音和损坏语音都是这样。图的下半部分显示 SPLICE 输出的$\hat{x}$ 参数以及变化量的范围（$\pm\sigma_x^2$）。

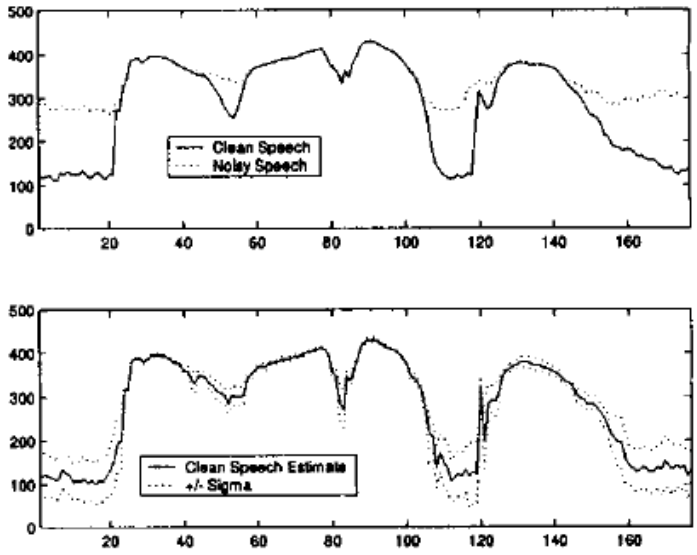

Fig. 2. C_0 **for clean speech (top), together with** $p(y|x)$ **for a corresponding noisy utterance, plotted as a function of** x**.**

倒频谱系数c_0大致和帧能量相关。在语言区域，语言的c_0太大了以至于覆盖了噪声的。$\hat{x}$的值是精确的，误差的边缘是很小的。在非语言区域，噪声下的c_0覆盖了语音的c_0，$\hat{x}$的值是不够精确的，并且伴随着很大的方差。我们从图 2 低半部分可以看出：干净语音的c_0是连续的，因为它的边界被虚线包围着。

4.2 定量分析

一些相关的数字实验是在 Aurora2[8]提供的语料库基础上，在框架上完成的。这个声学模型训练数据包含了 8440 个干净的语句以及同样的语料以 20 种不同的方式损坏，共分成 422 组。这 20 种方式包含四种不同的噪声类型（地铁环境，声音畸变，车载环境，展览环境），以及认为添加的 5 种不同的信噪比（无穷大，20，15，10 和 5）。我们所有的实验使用的带标记语言的干净数据都由 Aurora2 框架提供。

只有包含了 28028 个文件的集合 A 的实验结果在这篇文章中报告了。这个集合的噪声类型和训练数据中一样，但是信噪比被人为地添加了 7 种（无穷大，20，15，10，5，0 和-5）。

这篇文章中使用的 SPLICE 算法并没有很好的对隐藏的噪声类型（包含在 B 集合中）、隐藏的回旋信道（包含在集合 C 中）两种情况进行推广。尽管这些限制可以很轻松的被噪声均值标准化（NMN）和倒频谱举止标准化（CMN）所克服[4]，我们还是没有在这篇文章中采用这两种技巧。理论上来讲，集合 A 的结果是不会改变的，尽管我们没有考虑这两种技巧。

表 1 显示了 8 组不同实验的数字识别精度，包含了 3 种不同的前端和 3 种不同的解码策略。

Table 1. A comparison of the digit accuracy of eight front end configurations on Aurora test set A, with a clean acoustic model.

	Uncertainty Source		
Front End	None	SPLICE	True
Standard MFCC	63.66%	N/A	85.94%
SPLICE	87.22%	87.47%	89.52%
Complete SPLICE	88.21%	90.63%	92.81%

前端包含了三种方式，即标准 MFCC 算法、SPLICE 算法以及完全 SPLICE 算法。标准的 MFCC 算法只有一点和 Aurora 标准中的不同：我们使用的是内部的幅值平方谱而不是幅值谱。SPLICE 和完全 SPLICE 的后续处理在第三部分做了介绍。

我们考虑了三种情况的解码器：没有非确定性解码、基于 SPLICE 的非确定性解码和真实的 Oracle 数据库中得到的非确定性解码。基于 SPLICE 的非确定性解码器当然对于 MFCC 前端是不可能得到的，对于其他前端的计算结果已经在表中给出。真实的非确定性解码器的数据则来源于计算前端输出和真实的干净语言倒谱之间的幅值平方误差，这些数据都来源于 Aurora2 中的数据。

带有真实非确定性声学解码器的实验预示了通过使用非确定性声学解码我们可以得到多大的改进。对于标准前端，可以减少 60%的词错误率，仅仅是通过增加完全的倒频

谱的错误率的幅值。对于包含 SPLICE 的前端，可能的改进虽然小一些，但是也不容忽视。

使用基于 SPLICE 的非确定性声学解码在两方面都产生了改进。对于最好的显示系统，完全的 SPLICE 前端，包含非确定性声学解码，与不包含 SPLICE 的前端相比，可以减少词错误率达到 74.2%。

5、讨论

我们系统的阐述了如何将非确定性声学解码从噪声去除过程中整合到语音识别系统中。

非确定性声学解码带来了两大好处：一是：通过有效地包含条件概率 p(y|x)解释了剩余的损害来自噪声去除过程的原因。二是：因为非确定性声学解码是建立在广泛的概率框架上，我们避免了任何的启发式的调整。

应该很简单可以修改大多数的噪声去除算法已获得非确定性声学解码估计。我们已经通过 SPLICE 做到了。通过引入非确定性声学解码，我们可以在系统中减少词错误率达到 20%以上。

6、参考文献

[1]L.Deng,A.Acero,M.Plumpe,and X.D.Huang,"Large vocabulary speech recognition under adverse acoustic environments," in Proc.2000 ICSLP,Beijing,China,October 2000,pp.806-809.

[2]Y.Ephraim and M.Rahim,"On second-order statistics and linear estimation of cepstral coefficients," IEEE Trans.Speech and Audio Proc.,vol.7,no.2,pp.162-176,March 1999.

[3]Y.Zhao,"Frequency-domain maximum likelihood estimation for automatic speech recognition in additive and convolutive noises,"IEEE Trans. Speech and Audio Proc.,vol.8,no.3,pp.255-266,May 2000.

[4]J.Droppo,L.Deng,and A.Acero,"Evaluation of the SPLICE algorithm on the Aurora2 database(web update),"in Proc.Eurospeech 2001,Aalborg,Denmark,September 2001,pp.217-220.

[5]M.Cooke,P.Green,L.Josifovski,and A.Vizinho,"Robust automatic speech recognition with missing and unreliable acoustic data,"Speech Communication,vol.34,no.3,pp.267-285,June 2001.

[6]P.Green J.P.Barker,M.Cooke,"Robust ASR based on clean speech models:An evaluation of missing data techniques for connected digit recognition in noise,"in Proc.Eurospeech 2001,Aalborg,Denmark,September 2001,pp.213-216.

[7]J.Droppo,A.Acero,and L.Deng,"Efficient on-line acoustic environment estimation for FCDCN in a continuous speech recognition system," in Int.Conf.On Acoustics,Speech and Signal Processing,Salt Lake City,Utah,May 2001.

[8]H.G.Hirsch and D.Pearce,"The AURORA experimental framework for the performance evaluations of speech recognition systems under noisy conditions," in ISCA ITRW ASR2000"Automatic Speech Recognition:Challenges for the Next Millennium",Paris,France,September 2000.

附录C 录音内容

一、数字

一　/发音：一/

一　/发音：幺/

二　/发音：二/

二　/发音：两/

三

四

五

六

七

八

九

零　/发音：零/

零 /发音：栋/

十 /发音：十/

二十　/音：二十/

二、地名

蛇口新街

联合广场

蛇口沃尔玛

南玻大厦

白石洲

文化中心

世界之窗

锦绣中华

大梅沙

四海玫瑰园

市民广场

东门

体育馆

特区报社

沙头角

门诊部

投资大厦

北师大附小

桂庙新村

明华中心

海雅百货

小梅沙

海滨浴场

南山邮局

万科

三家店

海上世界

南油大厦

新时代广场

西丽医院

华强北

下沙

先进院

红树林

深圳大学

华侨城

会展中心

花果山

竹子林

康佳集团

深圳书城

兰溪谷

科技园

蛇口码头

市委

三、控制指令

打开空调

关闭空调

温度设定

打开雨刷

关闭雨刷

抬高座椅

放低座椅

打开车灯

关闭车灯

报告当前位置

地图

打开导航信息

关闭导航信息

显示油量

显示车速

拨打电话

暂停

停止

换台

收音机

调高音量

降低音量

播放 CD

怎么走

从

到

四、短语

以下三组指令的数字部分和地名部分随即产生，每次训练不一样。

1、随机指令，如：

打开车灯

降低音量

2、拨打电话 [Number]（5 句），如：

拨打电话 一 三 八

拨打电话 二 六 九 零

拨打电话 五 两 三 四 六 九 七

拨打电话 幺 三 八 七 六 九 五 六 六 七 幺

拨打电话 六 四 六 五 七 零 幺

3、温度设定 [Number] （5 句），如：

温度设定 一十四

温度设定 一十六

温度设定 二十三

温度设定 二十五

温度设定 十八

4、从[Place] 到[Place]怎么走（6 句），如：

从海上世界到先进院怎么走

从大梅沙到小梅沙怎么走

从蛇口沃尔玛到世界之窗怎么走

从四海玫瑰园到深圳书城怎么走

从市委到东门怎么走

从竹子林到康佳集团怎么走

在 学 取 得 成 果

致　　谢

值此论文完成之际，我深深的感谢我的老师、同学以及所有关心和支持我的人们，感谢中国科学院深圳先进技术研究院环绕智能实验室，感谢实验室给我提供的良好的学习环境和丰厚的技术积累，以及相关设备和数据资源，使我顺利完成这篇论文。

感谢北京科技大学的王沁教授，感谢她对我的特别关照和重视。

感谢中国科学院深圳先进技术研究院的蒙美玲教授和王岚博士，正是由于你们对我的重视才能让我提前赴先进院实习，完成这次本科毕业设计。您们严谨的治学态度、广阔的思路以及孜孜不倦的追求一直都在陶冶和熏陶着我，令我受益终身。

感谢先进院的邵阳师哥、李崇国师哥、欧阳建军师哥，邵阳师哥一直是我毕设的指导者，感谢你把我带入语音识别领域的大门；对于李崇国和欧阳建军，你们不仅仅在技术上给了我很多帮助，也在研究方向上给了我很多积极的建议。

最后感谢中国科学院深圳先进技术研究院在本次毕设中参与录音的 30 多位同学和老师，感谢你们的大力配合和积极支持。