多媒体会议中新型快速实时混音算法

王文林 廖建新 朱晓民 沈奇威
(北京邮电大学网络与交换技术国家重点实验室 北京 100876)

摘 要: 混音处理是多媒体会议系统中的一个关键环节, 直接影响用户之间的相互交流。现有常用的混音算法中存在着音量突变的问题, 通过对这些混音算法的分析, 得出了变化的混音权重是导致音量忽大忽小的主要原因的结论。在此基础上, 该文提出了一种采用与混音输入无关的恒定混音权重的非均匀波形收缩混音算法, 该算法混音结果自然流畅, 避免了音量突变的问题。该算法运算简单, 速度快, 没有乘除法操作, 容易硬件实现。可以广泛应用于大规模的多媒体会议系统中。

关键词: 多媒体会议; 音频处理单元; 非均匀; 波形收缩; 混音
中图分类号: TN919.8 **文献标识码:** A **文章编号:** 1009-5896(2007)03-0690-06

A Novel Fast Real-Time Audio Mixing Algorithm in Multimedia Conference

Wang Wen-lin Liao Jian-xin Zhu Xiao-min Shen Qi-wei
(*State Key Lab. of Networking and Switching Tech.*, *Beijing Univ. of Posts and Telecomm.*, *Beijing* 100876, *China*)

Abstract: In multimedia conference, audio mixing is an essential component, which affects the communication between users. At present, the commonly used audio mixing algorithms have a protean volume. By analyzing those algorithms, the conclusion of mutative mixing weights bring on protean volume is drawn. Base on this, a novel algorithm named Asymmetrical Wave-Shrinking (AWS) is proposed. A fixed mixing weight independent of inputs is used to ensure the natural and fluent outputs without protean volume. Without multiplication and division operations, the algorithm is so simple and fast that it can be easily implemented by hardware and widely applied in large scale multimedia conference systems.

Key words: Multimedia conference; APU (Audio Process Unit); Asymmetrical; Wave-shrinking; Audio mixing

1 引言

近年来, 多媒体会议成为多媒体通信发展的热点之一。在多媒体会议中, 音频交流最为频繁, 实时性的要求也远远高于视频及数据, 是多媒体会议中最基本的要素。为了具有更好的会议临场感, 与会者希望能同时听到多个发言者的声音。在分散控制会议模式下, 每个发言者的语音信号都单独传送给每个与会者, 在终端处进行混音后再播放, 这种方式需要占用大量的网络带宽, 影响语音信号的QoS(Quality of Service), 并对终端有较高的要求。为此, ITU-T提出了集中控制会议模式[1], 在MCU(Multipoint Control Unit)中对来自各发言者的语音信号进行混音处理, 再将结果传送到每个与会者, 极大地降低了网络传输的负担和终端的处理能力[2]。

目前各种混音算法都有其难以弥补的缺陷和不足, 很难满足大规模会议的应用。文献[3]中的平均混音算法随着混音路数的增加音量急剧降低; 文献[4,5]提出的对齐混音算法在混音过程中会出现明显的音量忽大忽小的变化; 文献[6]提出的自对齐混音算法则引入了部分噪音, 并且音量偏小; 实际应用较多的箝位混音算法[7]也存在音量突然变化情况。本文基于H.323 多媒体会议中集中式会议工作模式, 提出一种新型的快速实时混音算法, 采用与混音输入无关的恒定混音权重, 杜绝音量变化, 复杂度低, 基本不引入噪音, 且速度快, 极易于软硬件实现。

2 混音处理过程

根据ITU-T的H.323 规范, 在MCU中有MC(Multipoint Controller)和MP(Multipoint Processor)两大核心模块, 其中MP提供音频、视频和数据的集中处理能力, 划分为AMP(Audio MP), VMP(Video MP)和DMP(Data MP) 3 大模块。其中的AMP包括了多个APU(Audio Processing Unit), 分别对应一个会议, 各APU之间独立并行工作。APU的结构如图 1 所示[6]。

2005-08-18 收到, 2006-01-03 改回
国家杰出青年科学基金(60525110), 新世纪优秀人才支持计划(NCET-04-0111), 高等学校博士学科点专项科研基金(20030013006), 国家移动通信产品研究开发专项基金和电子信息产业发展基金资助课题

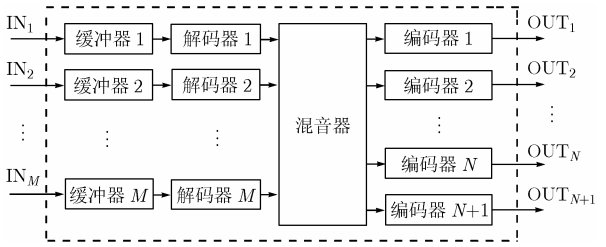


图 1　音频处理器结构图

在 H.323 集中控制模式下, 每个与会终端与 MCU 建立双工单播连接, 实时与 MCU 交换数据流。其中音频流的编码格式可以采用 G.711, G.722, G.723.1, G.728 和 G.729 编码规范, 与 AMP 中的某一个 APU 相连, APU 中的 Buffer 作为抖动缓冲使用, 可以在一定的程度上减少由传输导致的丢包、顺序不一致和抖动带来的影响。

经过缓冲的音频还必须经过相应解码器处理后才能由混音器进行混音, 混音完成后, 再根据输出终端不同采用相应编码器进行编码。所以, 混音一般以音频流的一帧为单位进行处理。

3　混音原理

原理 1　声音是由于物体振动对周围的空气产生压力而传播的一种压力波, 转换成电信号后, 再经过抽样、量化, 仍然是一种连续平滑的波形信号。

原理 2　量化后的语音信号的频率与声音的频率对应, 振幅与声音的音量对应。

原理 3　量化的语音信号的叠加等价于空气中声波的叠加。

所以当各信号的抽样率一致时, 混音可以实现为将各信号的采样数据线性叠加。在上述的 ITU-T 的 G.7xx 系列编码规范中, 只有 G.722 的抽样率是 16000, 其它规范的抽样率是 8000。如果要将 G.722 解码器的输出与其他解码器的输出混音, 只需要隔位丢弃采样数据即可。

假设在会议 v 中, 有 M 路音频参与混音, 在时刻 t 第 $i(i$=1,2 ,…, $M)$路音频解码输出的数据为 $a_i(t)$, 其值域为 $[-2^{Q-1},2^{Q-1}-1]$, 其中 Q 是量化精度。要求有 N+1 路编码输出, 通常 N=M。不失一般性, 可以约定第 $j(j$=1,2 ,…, $M)$路输出数据为 $b_j(t)$, 其中 $b_i(t)$ 是除 $a_i(t)$ 外其他 M–1 路的混音输出, 而 $b_{M+1}(t)$ 则是全部 M路的混音输出, 则

$$b_j(t)=\sum_{i=1,i\neq j}^{M}a_i(t),\ \ j=1,2,\cdots,M;\ \ b_{M+1}(t)=\sum_{i=1}^{M}a_i(t) \quad (1)$$

由式(1)可知, $b_j(t)$ 的值域不再是 $[-2^{Q-1},2^{Q-1}-1]$, 产生了溢出, 破坏了语音信号的特征参数, 从而引入了噪音。随着 M 的不断增大, 发生溢出的频率不断上升, 一般来说, 在 M=4 时, 溢出引入的噪音过大将导致混音后的语音无法辨认。所以, 混音算法的难点在于如何处理混音后采样值溢出问题。

4　已有混音算法分析

目前混音后采样值溢出处理方案较多, 但基本思想一致, 即根据原理 2 对语音信号的振幅做一定的平滑处理, 即可以在不破坏(或基本不破坏)语音信号原有的频率的基础上避免溢出, 故有

$$\left.\begin{aligned} b_j(t)&=\sum_{i=1,i\neq j}^{M}w_{i,j}(t)a_i(t),\ \ j=1,2,\cdots,M\\ b_{M+1}(t)&=\sum_{i=1}^{M}w_{i,M+1}(t)a_i(t)\end{aligned}\right\} \quad (2)$$

几乎所有的算法都由式(2)而来, 其中 $w_{ij}(t)$ 称为混音权重, 不同算法之间的区别往往仅仅是权重不相同。

(1) 平均算法　平均算法 (average audio mixing algorithm, 简称V算法)最为简单, 就是将采样数据线性叠加后取平均值[3], 所以其混音权重函数如下:

$$\left.\begin{aligned} w_{i,j}(t)&=\frac{1}{M-1},\quad i\neq j,\ \ i,j=1,2,\cdots,M\\ w_{i,M+1}(t)&=\frac{1}{M},\quad i=1,2,\cdots,M\end{aligned}\right\} \quad (3)$$

该算法的实质是将各路语音的音量减少了 M–1(或 M)倍, 所以随着 M 的增大, 各路语音的衰减将愈加严重, 最终导致语音细不可闻。而且, 随着会议的进行, 发言者数量 M 不断变化, 则 $w(x)$也不断变化, 导致音量忽大忽小甚至声音断续, 影响语音效果。所以 V 算法实际应用受到很大的限制, 无法适应大规模会议的混音需求。

(2) 对齐算法　文献[4]中的混音算法是一种典型的对齐算法(align audio mixing algorithm, 简称 A 算法), 该算法令各路音频流当前混音帧中采样值的绝对值的最大值为 TotalMax, 累加结果中采样值的绝对值的最大值为 MixedMax, 即

$$\left.\begin{aligned} \mathrm{TotalMax}_j&=\max_{\substack{t\in[T,T+\Delta t],i\neq j\\ i,j=1,2,\cdots,M}}\left(\left|a_i(t)\right|\right)\\ \mathrm{TotalMax}_{M+1}&=\max_{\substack{t\in[T,T+\Delta t]\\ i=1,2,\cdots,M}}\left(\left|a_i(t)\right|\right)\\ \mathrm{MixedMax}_j&=\max_{\substack{t\in[T,T+\Delta t]\\ j=1,2,\cdots,M+1}}\left(\left|b_j(t)\right|\right)\end{aligned}\right\} \quad (4)$$

其中 T 为该混音帧的起始时刻, Δt 为帧的长度。而混音权重如式(5), 其中 $L_j\in[1,\mathrm{MixedMax}_j/\mathrm{TotalMax}_j]$, 用来调整最终输出混音结果的值。但文献[4]没有考虑到 $\mathrm{MixedMax}_j$ 可能大于极限值 $2^{Q-1}-1$, 此时只能取 $L_j\in[1,(2^{Q-1}-1)/\mathrm{TotalMax}_j)$, 否则将导致溢出。

$$w_{i,j}(t)=\frac{\mathrm{TotalMax}_j}{\mathrm{MixedMax}_j}L_j,\ \ i=1,2,\cdots,M,\ j=1,2,\cdots,M+1 \quad (5)$$

文献[5]提出的混音算法与 A 算法类似, 但每一路语音有各自的权重, 所以能将低强度信号加强, 增大其可识别度,

但是也会将一些背景噪音放大, 其混音权重如式(6)所示。

$$w_{i,j}(t)=\frac{\max\limits_{t\in[T,T+\Delta t]}(|a_i(t)|)}{\text{MixedMax}_j},\quad i=1,2,\cdots,M,\ j=1,2,\cdots,M+1 \tag{6}$$

A 算法有着不可避免的缺陷。第一, 混音权重与各路音频采样值的最大值相关, 所以在计算时导致采样值需要处理两次。第二, 在实时混音过程中, 连续两个混音帧的混音权重不同, 这同样会导致音量忽大忽小。第三, 随着混音路数 M 的变化, 权重也不断变化, 导致音量变化。所以 A 算法并不能满足实时会议混音的需求。

(3) 箝位算法　箝位算法(pincers audio mixing algorithm, 简称P算法)引入了一个初始值为 1 的箝位因子 f_j [7], 混音权重表示为 $w_{i,j}(t)=f_j$, 其中$i\neq j$; i=1,2,…, M; j=1,2,…, M+1。

在计算过程中, 设 t_0 时刻 j 路输出值 $b_j(t_0)$ 发生溢出, 则立刻重新计算 $f_j=(2^{Q-1}-1)/|b_j(t_0)|$, 并将 $b_j(t_0)$ 设为饱和值。t_0 时刻以前的数据不再处理, 但 t_0 以后的计算使用 f_j 的新值。为了避免箝位因子 f_j 的值越来越小, 还必须在每隔 ΔT 时间检查 f_j 的值, 如果小于 1, 则将其微调至稍大一些的值, 即 $f_j=f_j+\Delta f$。在文献[7]中, ΔT 取一个采样时间, 但根据语音信号的特点可知, 高强度信号往往持续一段时间, 为了避免连续计算 f_j, ΔT 可以稍稍延长。

P 算法实现简单, 避免了 A 算法中采样值需要两次处理的问题, 音量忽大忽小和声音断续的问题也较对齐算法有所改善, 但如果在混音路数 M 较大, 则溢出的概率较大, 即 f_j 变化概率较大, 此时如果 M 增加, 则权重变化更加剧烈, 导致音量的变化十分明显。并且, 在 M 较大时, P 算法对弱信号也将衰减 f_j 倍, 将导致弱信号难以识别。所以 P 算法比较适用于小规模, 混音路数变化不大的会议中。

(4) 自对齐算法　文献[6]的自对齐算法(align-to-self audio mixing algorithm, 简称 AS 算法)提出以各路音频信号自身幅值所占的比例作为各自混音的权重, 参见式(7):

$$\left.\begin{aligned} w_{i,j}(t)&=|a_i(t)|\Big/\sum_{i=1,i\neq j}^{M}|a_i(t)|,\quad i\neq j,\ i,j=1,2,\cdots,M\\ w_{i,M+1}(t)&=|a_i(t)|\Big/\sum_{i=1}^{M}|a_i(t)|,\quad i=1,2,\cdots,M\end{aligned}\right\} \tag{7}$$

AS 算法不会发生溢出现象, 但是该算法混音时没有遵循线性关系, 所以会对语音频率产生一定影响而引入一定的噪音, 音量大小变化剧烈, 在参与混音的音频路数较多时表现尤为明显, 而且此算法比较复杂, 故不适合大规模实时会议中。

5　非均匀波形收缩混音算法

上述几个算法各有特点, 但缺陷大致相同。在 V 算法中, 权重随参与混音的路数 M 变化; 在 A 算法和 P 算法中, 权重随 TotalMax 以及 MixedMax 变化; 而在 AS 算法中, 权重变化更为剧烈, 随着每个采样值变化。因此上述各算法中的混音权重是时间 t 的函数。

若权重随时间变化, 则定会导致各音频流的音量随时间不同而有不同程度的缩放, 从而造成混音后音量大小变化不定, 这就是上述各算法各种缺陷的根本原因。

而解决这个问题的关键, 就是找到一个与时间 t 无关的权重。本文提出的非均匀波形收缩算法从这一关键问题着手, 尝试解决上述缺陷。

(1) 算法基本思想　混音路数 M 以及各路语音信号的强度 $a_i(t)$ 都是时间 t 的函数, 故混音权重与时间无关意味者它必须与混音路数 M 和 $a_i(t)$ 无关。此时无法根据 M 和 $a_i(t)$ 计算 $b_j(t)$ 的值域, 但由式(1)可知, M 和 $a_i(t)$ 取任意值时 $b_j(t)\in(-\infty,+\infty)$ 都成立, 所以在这里认为 $b_j(t)\in(-\infty,+\infty)$。

G.711 规范中采用分段量化规则是基于在语音信号中低强度信号比高强度信号出现几率更高的事实。非均匀波形收缩算法(Asymmetrical Wave-Shrinking audio mixing algorithm, 简称 AWS 算法)同样基于这一事实, 其基本思想与 G.711 类似, 采用分段收缩规则, 对线性叠加后的采样数值进行收缩来保证不出现溢出, 低强度信号采用较大的权重以确保信号的可识别性的同时获得一定的收缩比例, 而高强度信号时采用较小的权重以确保得到相应的收缩比例, 同时也保证一定的可识别性。

首先, 对欲收缩的问题域分段。因为 $b_j(t)\in(-\infty,+\infty)$, 则 $|b_j(t)|\in[0,+\infty)$, 将 $[0,+\infty)$ 均匀地划分为若干段, 每段长度为 2^{Q-1}, 有$[0,\ 2^{Q-1}],\cdots,((n-1)2^{Q-1},\ n2^{Q-1}]$, $(n2^{Q-1},\ (n+1)2^{Q-1}],\cdots$, 若$|b_j(t)|$落入区间$[(n-1)2^{Q-1},\ n2^{Q-1}]$, 则称其为 n 级强度信号。

其次, 确定各区间的收缩因子。引入基本收缩因子 $k(k>1)$, 并且遵循如下规定: 区间 0 内的收缩因子为 $(k-1)/k$; 区间 1 内的收缩因子为 $(1-(k-1)/k)[(k-1)/k]$; 区间 2 内的收缩因子则为 $(1-(k-1)/k)(1-(k-1)/k)[(k-1)/k]$; 依次类推, 区间 n 内的收缩因子为 $[(k-1)/k](1/k)^n$。

根据上述讨论, 混音权重 $w_{ij}(t)$ 与 i, j, t 均无关, 虽然不能直接表现为常数的形式, 但可定义为一个简单的映射关系: 令 $b_j(t)$ 为式(1)的结果, 则 $b'_j(t)=w(b_j(t))$, 如式(8)所示。其中 sgn(x)是符号函数, mod 是取余操作。

$$\left.\begin{aligned} b'_j(t)&=\text{sgn}(b_j(t))\sum_{i=0}^{n_j-1}\frac{k-1}{k}\left(\frac{1}{k}\right)^i 2^{Q-1}\\ &\quad+\frac{k-1}{k}\left(\frac{1}{k}\right)^{n_j}(b_j(t)\bmod 2^{Q-1}),\\ n_j&=|b_j(t)|\Big/2^{Q-1},\quad j=1,2,\cdots,M+1\end{aligned}\right\} \tag{8}$$

下面证明 $b'_j(t)$ 不会溢出。

证明　由式(8), 注意到n_j永远不能为 ∞, 而且$k>1$, 所以有

$$\left|b_j'(t)\right| = \left|\mathrm{sgn}(b_j(t))\sum_{i=0}^{n_j-1}\frac{k-1}{k}\left(\frac{1}{k}\right)^i 2^{Q-1} + \frac{k-1}{k}\left(\frac{1}{k}\right)^{n_j} \cdot\left(b_j(t)\bmod 2^{Q-1}\right)\right|$$

$$< \sum_{i=0}^{n_j-1}\frac{k-1}{k}\left(\frac{1}{k}\right)^i 2^{Q-1} + \frac{k-1}{k}\left(\frac{1}{k}\right)^{n_j} 2^{Q-1}$$

$$= \frac{k-1}{k}2^{Q-1}\sum_{i=0}^{n_j}\left(\frac{1}{k}\right)^i < \frac{k-1}{k}2^{Q-1}\sum_{i=0}^{\infty}\left(\frac{1}{k}\right)^i = 2^{Q-1}$$

即 $|\ b_j'(t)\ | < 2^{Q-1}$，因为采样值是整数，所以必有 $b_j'(t)\in[-2^{Q-1}, 2^{Q-1}-1]$，即不会发生溢出。　　证毕

(2) 算法实现与优化　在本算法的实际应用中，首要考虑基本收缩因子 k 的取值问题。为了运算方便，k 一般取 2 的整数次幂。根据算法的特点，k 值太小会对较大地收缩波形造成整体失真，太大则会导致高强度信号严重失真。所以取 k=8 或 16 较好，以下叙述中以 k=8 为例，此时该算法记为 8-AWS 算法。同时，根据 G.7xx 系列规范，取 Q=16。

下面计算 $[(k-1)/k](1/k)^n$ 的值。考虑一定的数据精度，不能展开所有的移位操作，所以有当 k=8 时:

$$7 = 4+2+1 \Rightarrow 7x = 4x+2x+x$$

$$= (x<<2)+(x<<1)+x \Rightarrow \frac{7}{8}\left(\frac{1}{8}\right)^{n_j} x$$

$$= ((x<<2)+(x<<1)+x) >> (3\times(n_j+1))$$

注意到 $\sum_{i=0}^{n_j-1}\frac{k-1}{k}\left(\frac{1}{k}\right)^i 2^{Q-1}$ 对于固定的 n_j，k 和 Q 来说是常数，并且当 n=5 时，$\frac{7}{8}\left(\frac{1}{8}\right)^5 2^{15}$=0. 875 已经小于 1，所以可以忽略掉 n_j >4 以后的数值。于是得到表 1。

由式(8)可知，只要计算出 n_j 之后，即可查表 1 得出 Rsh_{n_j} 和 K_{n_j}，可以求得混音结果，如式(9)所示。

$$\left.\begin{aligned} &b_{M+1}(t) = \sum_{i=1}^{M} a_i(t)\\ &b_j(t) = b_{M+1}(t) - a_j(t),\quad j = 1,2,\cdots,M\\ &n_j = \min\left(\left|b_j(t)\right| >> (Q-1), 4\right)\\ &c_j(t) = \left|b_j(t)\right| \,\&\, (2^{Q-1}-1)\\ &d_j(t) = (c_j(t)<<2)+(c_j(t)<<1)+c_j(t)\\ &b_j'(t) = \mathrm{sgn}(b_j(t))\left(K_{n_j} + (d_j(t) >> \mathrm{Rsh}_{n_j})\right) \end{aligned}\right\} \quad (9)$$

表 1　收缩因子累加式结果

n_j	$\mathrm{Rsh}_{n_j}=3\times(n_j+1)$	$K_{n_j}=\sum_{i=0}^{n_j-1}\frac{k-1}{k}\left(\frac{1}{k}\right)^i 2^{Q-1}$
0	3	0
1	6	28672
2	9	32256
3	12	32704
4	15	32760

其中没有特殊注明的 j 取值为 j=1, 2,…, M+1。所以，计算一路混音输出的算法流程模型如图 2 所示，此流程与其他各路混音互不干涉，具有较强的并行性。

所以，混音输入路数为 M 输出为 M+1 路时，按 abs(x), min(x, y), sgn(x) 3 个函数各为一次比较计算，一个采样值的混音运算的最大复杂度为 4M 次加法，M 次减法，4M 次移位，M 次按位“与”，M 次查表，3M 次比较操作，没有乘除操作，没有浮点运算，所以很容易采用硬件方式来实现。

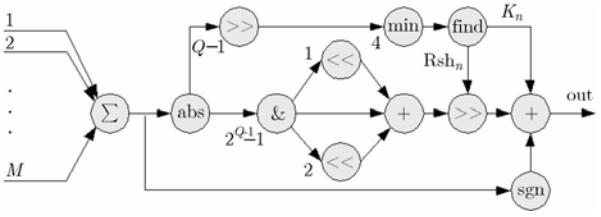


图 2　8-AWS 混音算法模型

6　实验结果

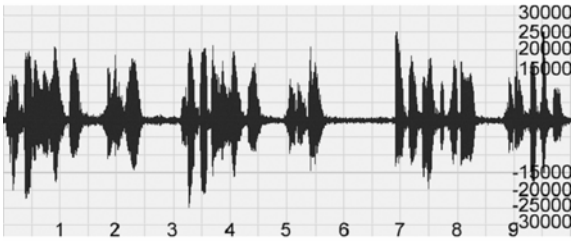
因为 P 算法与 A 算法的思想一致，并且前者明显优于后者，所以只选取 V 算法，P 算法，AS 算法以及 8-AWS 算法进行实验。在 P 算法中取 ΔT=10ms，Δf=0.05。

如图 3(a)，3(b)是由解码器解码后输入到混音器的两路语音，均为 10s，图 3(c)，3(d)，3(e)和 3(f)分别是由 V 算法，P 算法，AS 算法以及 8-AWS 算法混音后的输出。容易看出，V 算法的输出虽然没有丝毫改变原有波形，但是输出的波形都收缩了两倍，即音量随之降低一半。P 算法的输出波形比较符合原有波形，但当每次振幅溢出后，波形稍稍收缩，然后再慢慢扩大。而 AS 算法输出的波形失真较大，所有的波形尖锋明显被削弱甚至消失。8-AWS 算法的输出非常符合原有波形，但波形明显比 P 算法的小，因为该算法的基本收缩因子为 8，所有的波形都被至少收缩了 1/8。

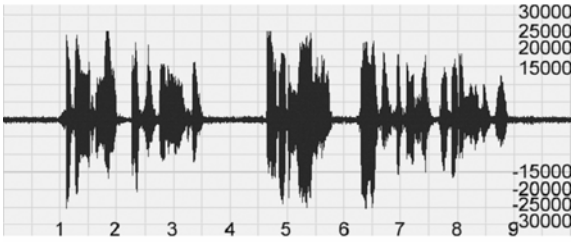
对图 3(c)，3(d)，3(e)和图 3(f)做的频谱分析，查看各波形的频谱分布情况。从原理 2 可知，均匀改变波形的振幅不会影响其频率，即 V 算法不会影响频谱分布。所以，在得到各频谱分布后，用图 3(d)，图 3(e)和图 3(f)的各频谱分布数值除以图 3(c)的相同频谱的分布数值，再对结果求均方误差，分别为 W_d=0.0120357，W_e=0.198118，W_f=0.0078736。

从上述结果看来，AS 算法输出的波形的频谱偏离较大，因为该算法采取自对齐的方式，每一次都会改变各自波形的混音权重；而 P 算法的混音权重在振幅溢出处会减少，然后慢慢增加，所以也出现了较大的偏离情况；8-AWS 的算法结果偏离最少，主要是因为该算法永远不会改变混音权重，只是采用非均匀收缩法，增加了高强度信号收缩比例。

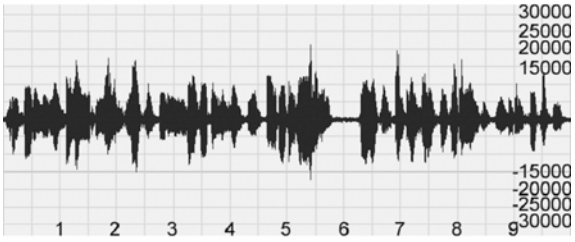
从主观的听觉感受而言，V 算法的结果音量偏小，随着混音输入的增多，音量越来越小，到 M=4 时已经无法分辨语音。而 AS 算法在输出大音量时语音不太自然，M=7 时已比较严重，但是还能识别出语音。P 算法的输出听觉效果较



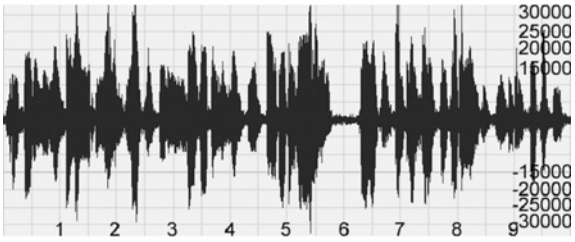
(a) 解码后的语音输入 1



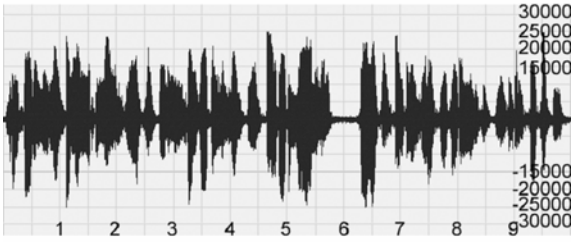
(b) 解码后的语音输入 2



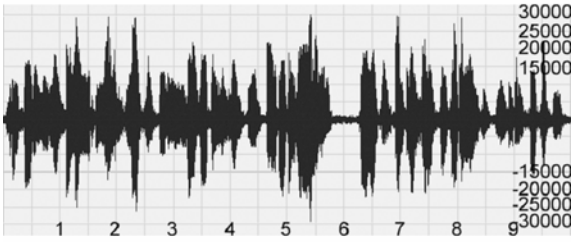
(c) V算法混音后输出



(d) P算法混音后输出



(e) AS 算法混音后输出



(f) 8-AWS 算法混音后输出

图 3 波形图

好, 没有明显的断续现象, 但是在突然的大音量后会有一定音量的减少, M=10 时感觉比较明显, 音量缓慢增加的过程基本感觉不出来; 当 M=5 时, 如果其中仅有一路信号强度较低, 则该路语音不能识别。8-AWS 算法输出语音的音量被减少成原有的 7/8, 但不影响声音辨认, 低强度语音信号也能轻易识别, 即使 M 增加到 15, 输出的语音一样自然流畅, 没有可觉察的噪音。

各算法的时间消耗的比较如图 4 所示。本次测试中输入为 10s 语音流, 每 10ms 为一帧, 共计 80000 个采样点, 执行 10 次, 取累加值, 即图中数值每个算法执行次数为 80 万次的时间。从图中可以看出, 参与测试的几种算法的时间消耗基本与混音输入数量成正比关系, 其中 AS 算法随输入增多而急剧上升, V 算法, P 算法以及 8-AWS 算法上升速度较为缓慢。在混音输入较少的时候, AS 算法要比 P 算法快, 在混音输入超过 22 路时, AS 算法的时间消耗超过了 P 算法。而 8-AWS 的算法速度和最简单的 V 算法非常接近, 比 P 算法快近 0.05s。

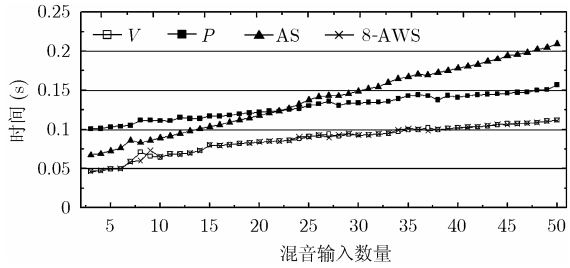


图 4 算法速度对比图

7 结束语

变化的混音权重导致混音后音量忽大忽小变化, 影响语音的识别。AWS 混音算法基于在语音信号中低强度信号比高强度信号出现几率更高的事实, 采用与混音路数无关的恒定混音权重, 混音效果理想, 混音后语音自然流畅, 没有噪音, 具有良好的主观听觉感受, 虽然音量比原有音量略微减小, 但是在基本收缩因子 k 取值恰当时丝毫不影响声音识别, 在多路语音输入时仍能保证语音质量。而且 AWS 算法简单, 实时, 快速, 是目前最快的混音算法之一, 完全满足多媒体会议中高性能, 高并发的混音要求, 将 AWS 算法与文献[8]中提出来的竞争策略结合后, 能支持大规模的混音应用。同时, AWS 算法不需要进行乘除法操作, 没有浮点运算, 容易采用硬件实现。

参 考 文 献

[1] ITU-T. Packet-based multimedia communication system. ITU-T Rec H.323 v4, 2000.

[2] Venkat R P, Harrick M V, and Srinivas R. Communication architectures and algorithms for media mixing in multimedia conferences. *IEEE/ACM Trans. on Networking*, 1993, 1(1): 20–30.

[3] Gonazález A J and Hussein A W. Audio mixing for interactive multimedia communications. JCIS'98, North

Carolina, 1998: 217–220.

[4] 杨树堂, 余胜生, 周敬利. 基于分组网络的多点实时语音混合及调度算法. 软件学报, 2001, 12(9): 1413–1419.
Yang Shu-tang, Yu Sheng-sheng, and Zhou Jing-li. A multipoint real-time speech mixing and scheduling algorithm based on packet networks. *Journal of Software*, 2001, 12(9): 1413–1419.

[5] 徐保民, 王秀玲. 一个改进的混音算法. 电子与信息学报, 2003, 25(12): 1709–1713.
Xu Bao-min and Wang Xiu-ling. An improved mixed audio algorithm. *Journal of Electronics & Information Technology*, 2003, 25(12): 1709–1713.

[6] 樊星, 顾伟康, 叶秀清. 多媒体会议中的快速实时自适应混音方案研究. 软件学报, 2005, 16(1): 108–115.
Fan Xing, Gu Wei-kang, and Ye Xiu-qing. Fast real-time adaptive audio mixing schemes in multimedia conferencing. *Journal of Software*, 2005, 16(1): 108–115.

[7] 马旋, 王衡, 汪国平, 等. 视频会议中混音后溢出问题的研究及解决方法. 第 13 届全国多媒体学术会议论文集, 宁波, 2004. http://graphics.pku.edu.cn/papers/2004.htm.

[8] 涂卫平, 胡瑞敏, 艾浩军, 等. 视频会议中音频多点处理器的研究. 武汉大学学报信息科学版, 2002, 27(1): 98–101.
Tu Wei-ping, Hu Rui-min, and Ai Hao-jun, *et al*.. Audio MP in video conference. *Geomatics and Information Science of Wuhan University*, 2002, 27(1): 98–101.

王文林：男，1979 年生，博士生，研究方向为多媒体通信、下一代网络增值业务.

廖建新：男，1965 年生，教授，博士生导师，主要研究领域为通信软件、增值业务提供技术.

朱晓民：男，1974 年生，博士，副研究员，主要研究领域为智能网、下一代业务网络、协议工程.

沈奇威：男，1976 年生，博士生，研究方向为网络智能、通信软件.