多媒体会议中的快速实时自适应混音方案研究*

樊 星+, 顾伟康, 叶秀清

(浙江大学 信息与电子工程学系,浙江 杭州 310027)

Fast Real-Time Adaptive Audio Mixing Schemes in Multimedia Conferencing

FAN Xing+, GU Wei-Kang, YE Xiu-Qing

(Department of Information and Electronic Engineering, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: Phn: +86-571-87951529, Fax +86-571-87951529, E-mail: starfan@mail.hz.zj.cn, http://www.zju.edu.cn

Received 2003-10-08; Accepted 2004-04-27

Fan X, Gu WK, Ye XQ. Fast real-time adaptive audio mixing schemes in multimedia conferencing. *Journal of Software*, 2005,16(1):108−115. http://www.jos.org.cn/1000-9825/16/108.htm

Abstract: In multimedia conferencing, multi-point controlling unit (MCU) provides the capabilities to process audio, video and data stream for multi-point conference. The capability of audio processing is basic and requires more for real-time criteria. This paper categorizes and analyzes the schemes, and a new multi-point speech audio mixing scheme using align-to-self weighted algorithm is provided to meet the demand of the practical need of multi-point speech processing. By applying the adaptive mixing algorithms, these high-performance processing schemes do not use the saturation operation which is widely used in multimedia processing. Therefore, no new noise will be added to the output, and they have low complexity and good hearing perceptibility. In the mean time, the schemes are designed for parallel processing, so they can be easily implemented with hardware, such as DSPs, and widely applied in multimedia conferencing systems.

Key words: multimedia conferencing; MCU (multi-point controlling unit); real-time; adaptive; audio mixing

摘 要: 多媒体会议中多点控制单元(multi-point controlling unit,简称 MCU)在多点会议中提供音频、视频和数据等的集中处理能力,其中音频处理能力是最基本的,也是实时性要求最高的要素.针对多点多媒体会议的实际应用需求,归类并分析了多种自适应多点语音混合处理方案,提出了采用自对齐加权的高性能混音方案.该方案不使用在实时多媒体处理中广泛运用的饱和运算,所以不引入新的噪声,因而具有较低的算法复杂度,其混合处理结果具有良好的听觉主观舒适感.同时,这套方案具有较好的并行处理特性,使用 DSP 等硬件较易实现,可以广泛应用在多媒体会议系统的实现中.

关键词: 多媒体会议;多点控制单元;实时;自适应;混音

中图法分类号: TP316 **文献标识码**: A

在今天的生活中分组网络已经非常普及.而作为分组网上的重要应用之一的实时多媒体通信业务也得到

* **作者简介**: **樊星**(1976−),男,重庆人,博士生,主要研究领域为图像处理,视频处理,音频处理,多媒体通信;**顾伟康**(1939−),男,教授,博士生导师,主要研究领域为计算机视觉,图像处理,模式识别;**叶秀清**(1936−),女,教授,主要研究领域为计算机视觉,图像处理,视频处理.

迅猛发展,诸多运营商都纷纷推出多媒体通信服务.在多媒体会议中,音频互动是基本的要素之一,它是多媒体会议中最基本的要素.由于在分组网络中没有 QoS(quality of service),所以网络的拥塞导致了端到端通信的语音丢包和延时抖动等问题[1];同时,多个端点同时相互发送数据更进一步增加了网络传输的负担,并且增加语音通信中数据收发的随机性和波动性.而语音互动的实时性要求远远高于多媒体会议中的其他要素[2],比如视频和数据.因为视频和数据在相对较长的时延内的抖动都是可以被用户接受的,而音频部分如果时间稍长,就会产生很明显的断续感,以致用户根本无法分辨语音所承载的语义从而严重影响沟通.为了解决这一问题,使用多点处理单元对语音信号进行混音,则降低了网络传输的负担,对于每个端点的处理能力的需求也大大降低[3].但是在常见的处理算法中,一般会因为多路语音信号采样量化数据叠加后超出量化上限[4],而导致不得不采用饱和运算将其变更为量化上限,这样就引入了新的噪声.本文则有针对性地提出一系列的解决方案.所提出的方案能很好地解决实际应用中的实时性、不引入噪声和保证合成分量音频特性的要求,增加单个服务端硬件平台的并发处理能力.

本文的应用背景是 H.323 集中式多点会议工作模式下的视频会议系统[5].所有的实验数据都是在自行开发的视频会议系统原型上进行实际实验得到的.

1 多媒体会议中音频多点处理器模型

根据国际电联(ITU)的推荐标准 H.323,多媒体会议服务器可以分割为两个核心模块:多点控制器(multipoint controller)和多点处理器(multipoint processor).多点处理器可以分为视频处理器(video MP)、音频处理器(audio MP)和数据处理器(data MP).其中 AMP 是我们研究的重点,它主要由音频编码器(audio encoder)、音频解码器(audio decoder)和混音器(audio mixer)构成.图 1 完整地描述了多点音频处理器的模型.

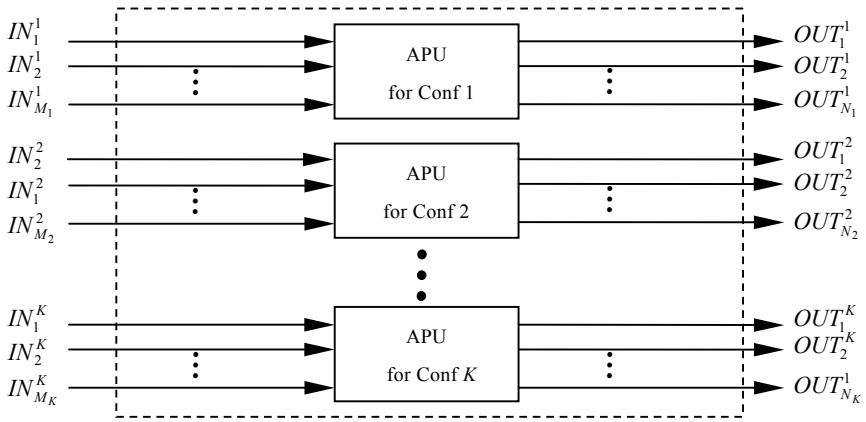

Fig.1 Model of audio MP

图 1 多点音频处理器的模型

从图中我们可以看出,一个 AMP 上可以同时支持相互完全独立的 K 个会议,每个会议对应一个独立的音频处理单元(audio processing unit,简称 APU),第 i 个音频处理单元分别有 M_i 个输入和 N_i 个输出.每个 APU 的结构如图 2 所示.

2 多媒体会议中的主要音频编解码规范

在集中式会议中,各个终端都与 MCU 建立基于单播(unicast)的连接,实时地向 MCU 发送和从 MCU 接收数据流.H.323 标准规定了在符合该标准协议族的系统中,语音编解码可以采用 G.711,G.722,G.723.1,G.728 和 G.729 这几套由 ITU 提供的建议标准.它们的相关属性见表 1.

混音器的输入均是各种编码方案解码后的数据,其输出为按照合成策略进行处理后的多路数据.

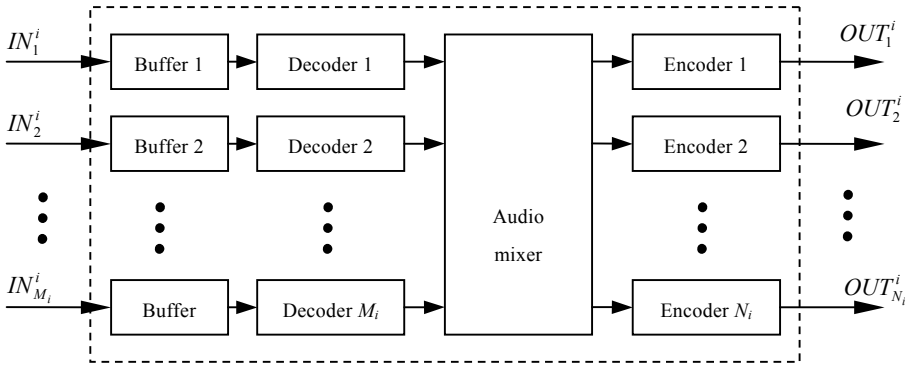

Fig.2 Model of APU

图 2 音频处理单元的模型

Table 1 Properties of audio codec in H.323 system

表 1 H.323 系统所用到的各种语音数据属性

Codec	Sampling rate (Hz)	Quantization (bits)	Target bitrate (kbps)	Channel
G.711	8 000	16	64/56	Mono
G.722	16 000	16	64/56/48	Mono
G.723.1	8 000	16	6.3/5.3	Mono
G.728	8 000	16	16	Mono
G.729	8 000	16	8	Mono

不失一般性,我们约定在 i 会议中有 M_i 个端点参与混音,在时刻 t ,第 $j(j=1,2,...,M_i)$ 路语音解码后的数据为 $a_j^i(t)$,其值域范围是 $[-2^{Q-1},2^{Q-1}-1]$,其中 Q 是采样量化位数.我们要求有 N_i+1 路输出,其中第 $k(k=1,2,...,N_i+1)$ 路输出的语音合成数据为 $b_k(t)$.通常情况下, $N_i=M_i$,则 $b_{N_i+1}(t)$ 为所有参与混音的端点的语音合成结果,而其他的都是除去本身的其余 N_i-1 路端点的混音结果.显然,经过合成可以得到 N_i 个 N_i-1 路合成结果和 1 个 N_i 路合成结果.合成结果的值域必须是 $[-2^{Q-1},2^{Q-1}-1]$,以便进行下一步的编码运算.如果实际计算结果超出,则需要进行溢出处理.

3 音频合成前后的处理

与会者的音频数据在经过编码后,从终端源源不断地发送至 MCU 的过程中,我们采用了 RTP[6]协议作为数据封装协议.但是由于网络传输中存在后发先至、数据包丢失等情况以及与会者本身作为音频信号发生源时其信号产生的统计特性不均匀等因素,导致了传输上的抖动和经传输后编码比特流的错误排序和数据缺失.为了很好地解决这个问题,一般采用抖动缓冲(jitter buffer)技术来加以解决,这也是图 2 中音频处理单元内的 Buffer 模块的主要功能.

经过缓冲处理后的数据必须经过声码器解码后才能参加混音,而混音后的数据必须经过编码后才能传回到相应的终端.

在本文提出的模型中,不要求使用同一套音频编解码方案,所以对于每一个音频处理单元,其输入、输出对应的编解码器可以根据需求自由组合选择.在这个意义上,本文描述的音频处理单元不仅具有混音功能,还具有转码(transcoding)的功能.

4 自适应音频混合方案

图 1 中的混音器是音频混合的核心模块.在实际应用中,如果与会人数只有两人,则只要保证通信是全双工的,就可以正常地进行会议,而无须进行混音.如果与会人数超过 3 人,则需要采用混音或者转发机制.

转发机制有两种策略:其一,将其他端点的数据都转发给一个端点;其二,按照约定的某种规则选出一路进行转发,也就是常见的“话筒传递”模式.这两种模式虽然可以满足一定层面的需求,但都存在明显的缺陷.前者会增加网络的传输负担和端点的处理负担,后者在多人会议的讨论中有明显的反应慢、效果差的缺陷.如果与会

者希望能够进行比较频繁的切换发言或者讨论,则会出现明显的断续和切换失效等情况.

而实时混音则能很好地解决这些问题.实际应用中,一般的混音方案都会采用时域叠加作为基本的处理手段.但是根据前面的分析可知,由于数字音频信号存在量化上限和下限的问题,则因叠加运算肯定会造成结果溢出.通常的处理手段是进行溢出检测,然后再进行饱和运算,即超过上限的结果被置为上限值,超过下限的值置为下限值.这种运算本身破坏了语音信号原有的时域特征,从而引入了噪声.这就是在某些系统中会出现爆破声和语音不连续现象的原因.同时,随着参与混音的人数增加,出现溢出的频率也不断上升,所以这类方法存在一个上限,而且这个上限值很低,实验证明,一般在 4 个终端参与混音时其结果就有很多噪音和断续,无法分辨语流了.

为了解决上述问题,我们引入实时自适应音频混合方案(real-time adaptive audio mixing scheme).在实际应用中,我们有 4 种基本的方案可以使用,它们是平均调整权重法(align-to-average weighted,简称 AAW)、强对齐权重法(align-to-biggest weighted,简称 ABW)、弱对齐权重法(align-to-weakest weighted,简称 AWW)和自对齐权重法(align-to-self weighted,简称 ASW).这 4 种方法都是采用加权平均的方式来进行处理的,一般地,我们定义 $a_j^i(t)$ 对应的权重为 $w_j^i(t)$,则我们可以得到一般的加权计算方法.

$$b_k^i(t)=\sum_{j=1,j\neq k}^{M_i} w_j^i(t)a_j^i(t)\Big/\sum_{l=1,l\neq k}^{M_i} w_l^i(t),\quad i=1,2,...,K;k=1,2,...,M_i;\quad b_{M_i+1}^i(t)=\sum_{j=1}^{M_i} w_j^i(t)a_j^i(t)\Big/\sum_{l=1}^{M_i} w_l^i(t) \tag{1}$$

平均调整权重法顾名思义,我们知道,只需要将权重取用相同的值即可,所以,我们取 $w_j^i(t)=1/M_i$,则根据式(1),我们有如下结果:

$$b_k^i(t)=\frac{1}{M_i-1}\sum_{j=1,j\neq k}^{M_i} a_j^i(t),\quad i=1,2,...,K;k=1,2,...,M_i;\quad b_{M_i+1}^i(t)=\frac{1}{M_i}\sum_{j=1}^{M_i} a_j^i(t) \tag{2}$$

这种方法虽然不引入噪声,但是由于对各个分量都进行了衰减[7],所以在与会数量上升时,混音合成的输出音量也随之下降.如果在参与合成的输入中有某一路或几路音量特别小,那么整个混音结果的音量会被拉低,一般来说,在 4 点以上的多点会议中使用这种方法,与会者都会感到合成后传回的声音较小,很多语音细节不能分辨.所以这种方法的实际应用也受到比较多的限制.

强对齐权重法是以参与的混音的输入按照其信号幅度值为依据来设计权重 $w_j^i(t)$.文献[1]中提出的方法可归入此类.参考文献[1]的方法,我们做了一定的更正.定义参与混音的所有 Buffer 块中数据的最大值为 $\overline{Total\mathrm{Max}}^i$,采用式(2)计算得到的 $b_k^i(t)$ 的最大值为 $\overline{Mixed\mathrm{Max}}^i$,则我们将会有,

$$b_k'^i(t)=b_k^i(t)\cdot\frac{\overline{Total\mathrm{Max}}^i}{\overline{Mixed\mathrm{Max}}^i}\cdot\mu^i \tag{3}$$

其中 μ^i 是一个调整因子,用以调整 $b_k'^i(t)$ 的幅值.它的取值应该是 $\left|\overline{Total\mathrm{Max}}^i\Big/\overline{Mixed\mathrm{Max}}^i\right|$ 的一个邻域,以混音缓冲区长度为限,对于混音结果做一个小范围的局部调整.这种方法有一定的自适应性.但我们容易发现,这种方法不可避免地存在量化溢出的问题.所以,在使用式(3)进行处理后,仍然需要进行溢出监测和饱和处理.虽然这种方法可以自动地将使用 AAW 方法得到的混音结果按照一个局部放大率进行放大,从而具有一定的自适应性,但是从全局来看,这种方法仍然存在引入噪声致使合成结果不理想的情况.这种方法还有一个比较大的缺陷就是,在某一路能量很低,另一路能量很高的情况下,其合成波形严重失真,具体的实验结果和分析见下面第 5 节.

弱对齐权重法则与上面所提方法相反,放大的依据主要是参与混音中的音量最弱项和采用某种特定方法合成输出的最弱项.这种方法的主要好处是能够将参与会议的声音较弱端点的声音放大,使得其语音细节可以被辨识.但是这种方法和 ABW 一样,存在着溢出检测和饱和处理的问题.

自对齐权重法则是考虑参与混音的多路音频信号自身的特点,以它们自身的比例作为权重,从而决定它们在合成后的输出中所占的比重.我们定义:

$$w_j^i(t)=\left|a_j^i(t)\right|\Big/\sum_{p=1}^{M_i}\left|a_p^i(t)\right| \tag{4}$$

将式(4)代入式(1),有

$$\left.\begin{aligned} b_k^i(t)&=\sum_{j=1,j\neq k}^{M_i}\left(\left|a_j^i(t)\right|\cdot a_j^i(t)\Big/\sum_{p=1,p\neq k}^{M_i}\left|a_p^i(t)\right|\right)\Big/\sum_{l=1,l\neq k}^{M_i}\left(\left|a_l^i(t)\right|\Big/\sum_{p=1,p\neq k}^{M_i}\left|a_p^i(t)\right|\right),\quad i=1,2,...,K;k=1,2,...,M_i\\ b_{M_i+1}^i(t)&=\sum_{j=1}^{M_i}\left(\left|a_j^i(t)\right|\cdot a_j^i(t)\Big/\sum_{p=1}^{M_i}\left|a_p^i(t)\right|\right)\Big/\sum_{l=1}^{M_i}\left(\left|a_l^i(t)\right|\Big/\sum_{p=1}^{M_i}\left|a_p^i(t)\right|\right)\end{aligned}\right\}\tag{5}$$

经过化简和整理后,我们有:

$$\left.\begin{aligned} b_k^i(t)&=\sum_{j=1,j\neq k}^{M_i}\left(\left[a_j^i(t)\right]^2\cdot\mathrm{sgn}\left[a_j^i(t)\right]\right)\Big/\sum_{l=1,l\neq k}^{M_i}\left|a_l^i(t)\right|,\quad i=1,2,...,K;k=1,2,...,M_i\\ b_{M_i+1}^i(t)&=\sum_{j=1}^{M_i}\left(\left[a_j^i(t)\right]^2\cdot\mathrm{sgn}\left[a_j^i(t)\right]\right)\Big/\sum_{l=1}^{M_i}\left|a_l^i(t)\right|\end{aligned}\right\}\tag{6}$$

其中 $\mathrm{sgn}(x)=1,x>0;\mathrm{sgn}(x)=0,x=0;\mathrm{sgn}(x)=-1,x<0$.并且我们约定参与混音的输入能量之和不为 0,这就保证了上式分母不可能为 0.如果参与混音的各路输入的能量之和为 0,就说明各路输入的幅值为 0,也就不需要进行混音了.

结合上面的分析,我们提出一种处理结构,如图 3 所示.从图中容易看出,这个混音方案已经进行了很好的优化,整个混音的实现过程在图中得到了明确的表述.某一路输入 $a_j^i(t)$ 首先用以计算 $\overline{SG}_j^i(t)=\mathrm{sgn}\left[a_j^i(t)\right]$,利用这一结果计算 $\overline{AB}_j^i(t)=\left|a_j^i(t)\right|=a_j^i(t)\cdot\overline{SG}_j^i(t)$,然后再次利用前面结果计算 $\overline{SQ}_j^i(t)=a_j^i(t)\cdot\overline{AB}_j^i(t)$.所计算的各路分量的两个中间结果分别用以计算 $\overline{SAB}^i(t)=\sum_{j=1}^{M_i}\overline{AB}_j^i(t)$ 和 $\overline{SSQ}^i(t)=\sum_{j=1}^{M_i}\overline{SQ}_j^i(t)$.这两者相除可以得到全部输入的混音结果.在后续计算中,从 $\overline{SAB}^i(t)$ 和 $\overline{SSQ}^i(t)$ 去掉自己这一路输入所对应的值,然后相除则可以得到对应于某一路输入的经过回声抑制后的混音输出.

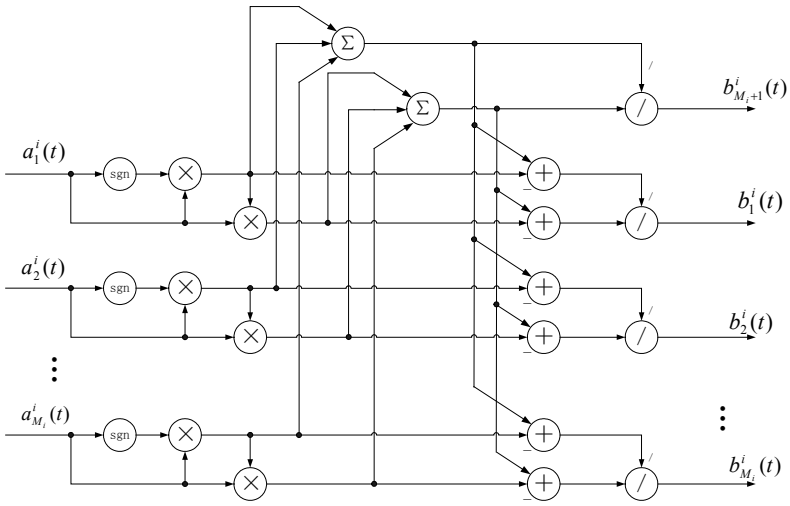

Fig.3 Framework of ASW model

图 3 ASW 混音器模型示意图

图 3 所示的结构具有良好的并行处理特点,根据这个处理模型,我们可以很容易地使用 DSP 等硬件来实现其功能.如果在 Intel IA32 架构的平台上,我们可以很方便地使用 MMX/SSE/SSE2 指令集对系统算法进行优化,并可以获得更高的计算效率,从而进一步提高会议的并发处理能力和实时性.

5 实验结果

我们选取有代表性的两种算法进行实际实验,一种是根据文献[1]改进的 ABW 方法实现,另一种是本文提出的 ASW 方法实现.首先我们从合成后音频输出的波形上来分析合成效果及其听觉感受.

我们比较图 4(c)和图 4(d)的波形,容易看出,当两路输入中有一路很弱而另一路较强时,ASW 方法的混音结果明显优于 ABW 的结果.从整体上看,ABW 输出的波形毛刺明显多于 ASW 的输出.从波形的包络上看,ASW 输出波形的包络明显地与两路输入的包络叠加更为接近;而 ABW 输出波形的包络则变形较大,这正是由于局部缩放因子引起的不良结果.

根据实际参与测试的测试人员的主观评价,ASW 输入的效果明显好于 ABW 的输出.ASW 合成后的音频流连续、自然,没有跳音和断续的感觉,也没有爆破噪声.ABW 合成后的音频流虽然很少有爆破噪声,但是有比较大的空洞感,同时由于分段缩放的因子取值各有不同,所以输入音频流会产生突然性的音量大小变化.综上,我们可以看出本文提出的 ASW 模型明显改善了文献[1]提出的 ABW 模型的缺陷.

(a) Decoded audo input 1

(a) 解码后的语音输入 1

(b) Decoded audo input 2

(b) 解码后的语音输入 2

(c) Two stream mixed output by ABW

(c) ABW 方法两路混音处理后的结果

(d) Two stream mixed output by ASW

(d) ASW 方法两路混音处理后的结果

Fig.4

图 4

我们的具体测试方法是,通过更改参与混音人数来测试不同混音路数的混音器实时运行特性.实验时,语音数据取 1 000 帧,每帧 120 个采样点(240 Bytes),共执行 10 次,总时间长度为 150s.每次运行 Intel(R) VTune(TM) Performance Analyzer 7.0,在最后的输出结果中取语音混合函数的“Self Time”和“Total Time”两个指标,将结果记录下来,并计算出平均每个采样点所消耗的时间.

从实验结果上可以看出,随着混音路数的增加,每个采样点平均消耗时间的增加很缓慢,而且非常接近一个

混音单元的计算时间消耗.如图 5 所示.

图 5 中横轴表示混音器和混音单元中的混音路数,纵轴表示混音器和混音单元的时间消耗(μs),从图中我们可以看出,混音路数和混音的时间消耗呈近似线性关系;而且采用的两种方法的性能相差很大.随着混音路数的增加,采用 ASW 模型的混音方案的性能改善也越发明显,这表明,基于 ASW 合成模型的方案完全适用于人数较多的混音环境.

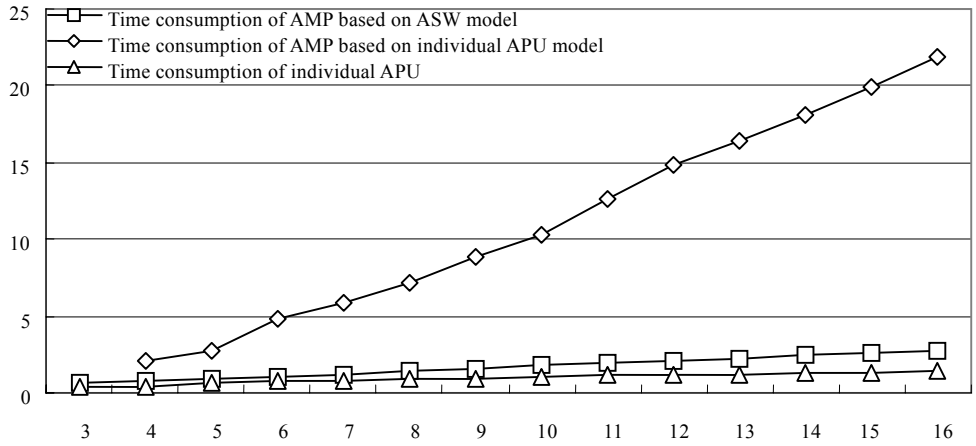

Fig.5 Comparison of test results

图 5 测试数据比对图示

表 2 中的算法是采用独立的混合单元建立的混音器模型,我们可以看出,其时间消耗随着参与混音点数的增加而急剧上升.我们再看采用图 3 所示的 ASW 模型实现的混音器的测试结果,见表 3.容易看出,采用 ASW 模型实现的混音器的时间消耗随着参与混音点数的增加上升很平缓.

Table 2 Time consumption of AMP based on individual APU model

表 2 基于独立混音单元模型实现的混音器的时间消耗

Input num.	Run times	Frames	Samples	Call times	Selfs time (s)	Total time (s)	Comsumption of APU per sample (s)	Comsumption of AMP per sample (s)
3	10	1 000	120	10 000	64 213	487 297	0.406 080 833	
4	10	1 000	120	10 000	66 298	499 533	0.416 277 5	2.040 600 832
5	10	1 000	120	10 000	71 312	813 385	0.677 820 833	2.759 208 333
6	10	1 000	120	10 000	72 335	861 305	0.717 754 167	4.784 679 165
7	10	1 000	120	10 000	73 917	949 241	0.791 034 167	5.815 313 336
8	10	1 000	120	10 000	80 750	1 051 979	0.876 649 167	7.204 922 503
9	10	1 000	120	10 000	81 857	1 108 437	0.923 697 5	8.813 540 003
10	10	1 000	120	10 000	86 058	1 253 319	1.044 432 5	10.281 407 5
11	10	1 000	120	10 000	94 193	1 362 439	1.135 365 833	12.624 123 33
12	10	1 000	120	10 000	94 355	1 398 307	1.165 255 833	14.789 645 83
13	10	1 000	120	10 000	90 438	1 448 525	1.207 104 167	16.355 43
14	10	1 000	120	10 000	98 889	1 490 533	1.242 110 833	18.141 569 17
15	10	1 000	120	10 000	97 100	1 539 408	1.282 84	19.914 502 5
16	10	1 000	120	10 000	100 228	1 669 526	1.391 271 667	21.916 711 67

Table 3 Time consumption of AMP based on ASW model

表 3 基于 ASW 模型实现的混音器的时间消耗

Input num.	Run times	Frames	Samples	Call times	Selfs time (s)	Total time (s)	Comsumption of APU per sample (s)
3	10	1 000	120	10 000	228 399	783 174	0.652 645
4	10	1 000	120	10 000	284 890	942 969	0.785 807 5
5	10	1 000	120	10 000	339 689	1 150 032	0.958 36
6	10	1 000	120	10 000	392 104	1 255 496	1.046 246 667
7	10	1 000	120	10 000	415 897	1 395 979	1.163 315 833
8	10	1 000	120	10 000	498 878	1 738 713	1.448 927 5
9	10	1 000	120	10 000	545 231	1 919 448	1.599 54
10	10	1 000	120	10 000	615 581	2 118 067	1.765 055 833
11	10	1 000	120	10 000	671 477	2 322 645	1.935 537 5
12	10	1 000	120	10 000	729 800	2 482 225	2.068 520 833
13	10	1 000	120	10 000	791 654	2 667 262	2.222 718 333
14	10	1 000	120	10 000	856 685	2 999 607	2.499 672 5
15	10	1 000	120	10 000	899 598	3 161 552	2.634 626 667
16	10	1 000	120	10 000	955 423	3 347 876	2.789 896 667

6 结 论

综上所述,不仅在一般的应用场合,基于 ASW 合成模型的混音方案能够满足实际需求,而且在具有高并发量要求的混音时使用ASW模型也能获得高质量的实时混音结果.它不仅保证了在多点混音时的高性能,具有很高的实时性,同时它也保持了参与混音的各路输入的时域细节特征,因而具有很好的听觉主观舒适感和连续感.相对于近年来提出的一些语音混合算法[1,7],算法性能和输入效果都有明显的改善.

References:

[1] Yang ST, Yu SS, Zhou JL. A multipoint real-time speech mixing and scheduling algorithm based on packet networks. Journal of Software, 2001,12(9):1413−1419 (in Chinese with English abstract).

[2] Daigle JN, Langford ID. Model for analysis of packet voice communications systems. IEEE Journal on Selected Areas in Communications, 1986,4(6):847−855.

[3] Venkat RP, Harrick MV, Srinivas R. Communication architectures and algorithms for media mixing in multimedia conferences. IEEE/ACM Trans. on Networking, 1993,1(1):20−30.

[4] Agustín JG, Hussein AW. Audio mixing for interactive multimedia communications. In: Wang P, ed. Proc of the JCIS'98. NC: Research Triangle, 1998. 217−220.

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

[6] Schulzrinne H, Caner S, Frederick R, Jacobson V. RTP: A transport protocol for real-time applications. IETF RFC 1889, IETF, 1996.

[7] Tu WP, Hu RM, Ai HJ, Xie X. Audio MP in video conference. Geomatics and Information Science of Wuhan University, 2002, 27(1):98−101 (in Chinese with English abstract).

附中文参考文献:

[1] 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.

[7] 涂卫平,胡瑞敏,艾浩军,谢兄.视频会议中音频多点处理的研究.武汉大学学报(信息科学版),2002,27(1):98−101.