Integrating a HRTF-based Sound Synthesis System into Mumble

Martin Rothbucher, Tim Habigt, Johannes Feldmaier and Klaus Diepold

Institute for Data Processing, Technische Universität München
Arcisstr. 21, 80333 München, Germany
{martin.rothbucher, tim, johannes.feldmaier, kldi}@tum.de

***Abstract*—This paper describes an integration of a Head Related Transfer Function (HRTF)-based 3D sound convolution engine into the open-source VoIP conferencing software *Mumble*. Our system allows to virtually place audio contributions of conference participants to different positions around a listener, which helps to overcome the problem of identifying active speakers in an audio conference. Furthermore, using HRTFs to generate 3D sound in virtual 3D space, the listener is able to make use of the cocktail party effect in order to differentiate between several simultaneously active speakers. As a result intelligibility of communication is increased.**

I. Introduction

Nowadays, economic and social developments require more and more interactive communication over teleconferencing systems to manage work progress and projects, or to just communicate while playing network computer games.

Recently, there was great progress concerning the quality of speech by increasing bandwidth of internet connections and by implementing capable algorithms to suppress noise. Furthermore, computer generated visualizations of conference rooms and the conference participants have been implemented to increase the degree of immersion [1].

High quality but very expensive solutions for professional teleconferencing are available on the market. In these systems each conference attendant at a remote site is recorded by a camera and is shown on a screen in the conference room. The sound is recorded by a microphone and played below the screen showing the participator. In such a way, a high degree of audio immersion is expected to be achieved, because the audio signals are in fact played at physically different places in the conference room.

Such a system is usually too expensive for small companies and computer gamers. As a simple solution, most of them are using freeware to conduct teleconferences. One popular teleconferencing freeware is *Skype*, using direct peer-to-peer connection between the users. Audio codecs used by *Skype* seek to improve speech quality but do not use binaural techniques to generate 3D sound. Another prominent conferencing tool for gamers is *Teamspeak*. Unfortunately, neither *Skype* nor *Teamspeak* are open source.

MMSP'10, October 4-6, 2010, Saint-Malo, France.
978-1-4244-8112-5/10/$26.00

Recently *Ekiga*, an open source VoIP software, was equipped with binaural sound rendering to make use of the well-known cocktail party effect [2], enabling the user to virtually differentiate and understand simultaneously talking speakers [3]. *Ekiga* uses the Session Initiation Protocol (SIP) to initiate direct connections between each user. These connections are symmetric with respect to their up- and download bandwidth. If there are lots of participants or the upload is restricted, this might result in a loss of quality or the number of participants has to be limited.

In our approach, we use the open-source voice chat software *Mumble*. *Mumble* has a vivid and active community which consequently pushes the development and quality of the software. *Mumble* is using proprietary Transmission Control Protocol (TCP) messages and User Datagram Protocol (UDP) packets to communicate with the server component *Murmur*. All traffic is routed through the server and only one outgoing connection is needed. This results in an asymmetric network load, which is advantageous for the common network architecture. Another advantage of *Mumble* is the well-arranged structure and the consequent adjustment towards voice communication. Also, the source code is easy to understand and portable, using C++ and the Qt framework.

In this paper, we give an introduction to Head Related Transfer Function (HRTF)-based sound synthesis and describe how to integrate our approach into *Mumble*.

The paper is organized as follows. Section II provides basic knowledge about HRTF data. In section III, different sound convolution techniques are summarized. Section IV explains the structure of *Mumble* our approach of integrating HRTF sound convolution into *Mumble*. Finally the 3D audio extension is presented in section V and a conclusion is given in section VI.

II. Head Related Transfer Functions

The human head, shoulder, torso and pinna can be regarded as a linear and time invariant system for sound waves as they enter the ear canal [4]. Due to reflexion and diffraction by the human body, sound waves have to undergo spectral changes on their way to the ear drum, described by the so called HRTFs. Since the diffractions and reflexions differ from direction to direction, HRTFs can be considered as direction dependent filters. It is important to know that a set of HRTFs is unique

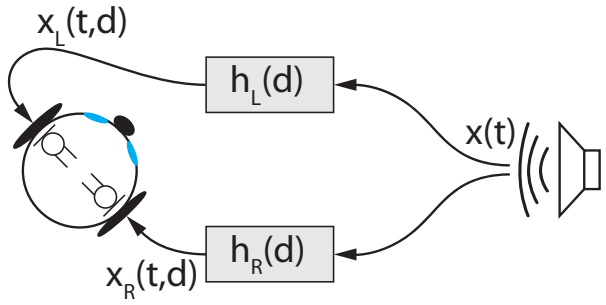

Fig. 1. Measurement of the direction dependent filters

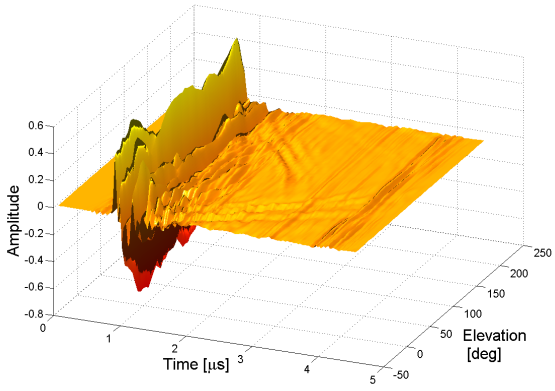

Fig. 2. Example of HRIRs in the saggital plane

for each individual due to unique geometric features of each person.

Usually, a set of HRTFs is generated by measuring the Head Related Impulse Responses (HRIRs), the time domain representation of the HRTFs, by a time-consuming procedure in an anechoic chamber [5]. Small microphones are placed in the entrance of sealed ear canals and record sound presented by a loudspeaker at a certain position. Then the loudspeaker is moved to the next position and the procedure is repeated. Depending on the density of the sampling grid, the process of measuring a set of HRTFs can last several hours.

Figure 1 illustrates a setup to measure HRIRs. The mono signal x is unequally delayed before arriving at the ears due to path differences, which lead to the so-called interaural time difference (ITD). Additionally, the loudness of the signals is influenced by shadowing effects of the head, resulting in interaural level differences (ILD). Finally, diffraction and reflexion from the listener's body are changing the spectrum of the signal. The recorded left and right signals x_L and x_R can be described by

$$x_L(t,d) = h_L(d) * x(t) \tag{1}$$

and

$$x_R(t,d) = h_R(d) * x(t), \tag{2}$$

where d is the direction of the sound source relative to the ears and $*$ describes the convolution operation in time domain.

Figure 2 shows a set of recorded impulse responses of the right ear in the saggital plane. It initiates that direction dependent reflexions and diffractions by the body lead to different responses.

With a set of measured HRTFs it is possible to convert a mono signal to a 3D presentation via stereo headphones. Computing (1) and (2), one can generate the spatial sound signals $x_R(t,d)$ and $x_L(t,d)$ by convolving a mono signal $x(t)$ with $h_L(d)$ and $h_R(d)$, respectively.

III. Convolution Techniques

To add direction information to a mono signal using a HRTF based approach, it is necessary to filter the input signals with the characteristic HRTFs. For every position in the virtual space there are two transfer functions $h_L(d)$ and $h_R(d)$ which have to be convolved with the input signal $x(t)$.

A. Convolution in Time-Domain

Several algorithms exist to perform convolution. Obviously, all of them have their advantages and disadvantages. The most straightforward algorithm calculates the convolution in time-domain.
The definition of discrete-time convolution is given by

$$(x * h)[n] = \sum_{m=-\infty}^{\infty} x[m]\, h[n-m], \tag{3}$$

where x denotes the input signal and h denotes the impulse response of the filter. This algorithm has the favorable property that it does not introduce additional latency. On the other hand, the computational cost is relatively high.
For every output sample, the algorithm for convolution in time-domain has to carry-out M multiplications and additions, where M is the length of the FIR-filter h.

B. Fast Convolution Techniques

Convolution of two signals can be computed much more efficiently using the fact that a convolution in time-domain corresponds to a simple multiplication in frequency-domain. This property is exploited in the commonly-used *Overlap-Add* and *Overlap-Save* algorithms. These algorithms reduce the computational complexity but introduce an inherent delay compared to convolution approaches in time-domain.
The delay introduced by filtering in the frequency-domain originates in the blockwise computation of the Discrete Fourier Transform (DFT). A whole DFT-block has to be buffered in the signal processing system before the actual computation can begin.

This leads to a miniumum time-delay d of

$$d = \frac{L}{F_s}, \tag{4}$$

where L is the length of a processing block and F_s denotes the sample rate of the sound synthesis.

Figure 3 illustrates the operation of the Overlap-Add algorithm. The input stream is divided into several blocks of

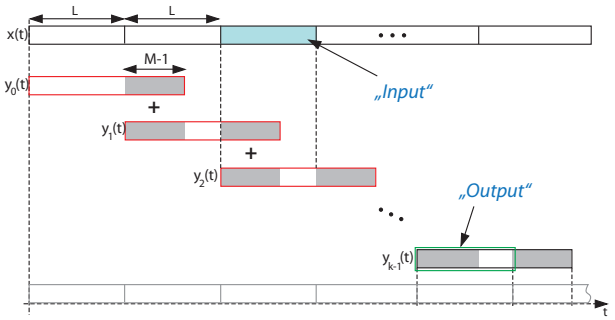

Fig. 3. Scheme of the Overlap-Add algorithm.

an arbitrarily chosen length L. These blocks are zero-padded with $M-1$ samples. Each of these blocks is transformed into frequency domain via DFT. There they are multiplied with the Fourier Transform of the filter coefficients (note that these have to be calculated only once). After the Inverse Fourier Transform, the results are stored in $y_i(t)$.
These output blocks consist of more samples than the input blocks, which stems from the fact that a linear convolution produces longer results than the circular convolution calculated by the DFT.
To obtain the output signal of the linear convolution, overlapping parts of the result blocks $y_i(t)$ have to be added.
A drawback of these algorithms is the direct relation of the filter lengths to the delay. The length of the DFT-blocks has to be at least as big as the length of the filter. To use the computational advantages of the fast fourier transform (FFT) this block length is generally even higher (as FFT block-lengths are powers of 2).

C. Partitioned Fast Convolution

To reduce the delay of fast convolution techniques it is advantageous to have very short filters. A comparably simple but nevertheless computationally efficient algorithm to combine the advantages of fast convolution with low delays was introduced by Torger and Farina in 2001 [6]. This algorithm cuts a long impulse response into several shorter partitions. Although this algorithm is theoretically less efficient than an earlier published partitioned convolution algorithm by Gardner [7], it performs very well in real implementations and has the benefit of being free from patent restrictions.

IV. Integration of 3D Audio into Mumble

This section describes the structure of *Mumble* concerning the implementation of a HRTF 3D sound convolution engine into the teleconference system.

A. Structure

Figure 4 shows the schematic structure of the *Mumble* modules. The GUI is the project's core, as common for Qt applications. A server handler provides methods for communication with the server. The audio samples are transmitted using

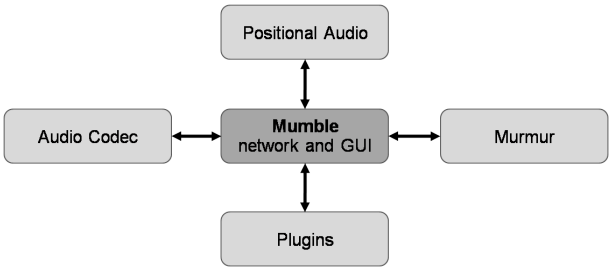

Fig. 4. Structure of Mumble modules

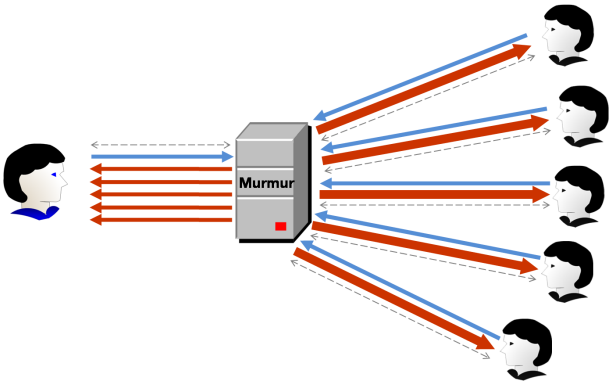

Fig. 5. Mumble conference using a server

UDP packets after establishing a TCP control connection to the server. As shown in Figure 5, *Mumble* only creates connections between the server component and the user, leading to an asymmetric network load, which is advantageous regarding the existing network structure.

Another module is *Murmur*, a stand-alone server application for *Mumble*. *Murmur* provides a high functionality such as access control methods (ACL, Access Control Lists). Without exception, the connections between *Mumble* and *Murmur* are encrypted with TLSv1 AES256-SHA for the control connection and OCB-AES128 for the audio streams.

The *Plugins* module enables developers to quickly integrate add-ons into the software by controlling the coordinates used for the calculation of relative positions of the connected users. The position data is used for choosing the HRTF corresponding to the position of the conference participants.

The speech coding module (*Audio Codec*) in Figure 4 compresses the audio data. After the data is encoded, it is transfered in segments over UDP. The received packets are processed in the *Positional Audio* module. *Mumble* has actually been developed to allow communication without time lag between persons playing online games. The *Positional Audio* module virtually places communication partners corresponding to the position of their avatar by just multiplying the audio signal with an attenuation coefficient [8]. In this way, a taste of 3D audio can be experienced by creating different sound volume levels with a 5.1 speaker system.

We extended the *Positional Audio* module with our 3D sound convolution engine, enabling 3D sound rendering with HRTFs.

B. HRTF Integration

To enable HRTF sound synthesis for *Mumble*, we identified the *AudioOutput* class. This class provides all main methods for audio processing. The simplified class diagram, shown in Figure 6, summarizes the important parts, already extended with our classes of the HRTF filter. Within the *AudioOutput* class, the method *mix* merges the incoming audio streams of active conference participants by directly adding up the signals or adding them after a previous weighting with an attenuation coefficient (*Positional Audio Mode*). For HRTF sound rendering, we extend *mix* by implementing a filter that applies different HRTFs on the incoming audio streams.

Our HRTF rendering can be regarded as a virtual transmission path, shown in Figure 7. The audio streams of different speakers are virtually attached to a HRTF-Filter module, which has to have properties of a sink, filter and source that are inherited from *ISource*, *IFilter* and *ISink* as shown in Figure 6. The HRTF-Filter module has also to be attached to the *AudioOutput* class, which has inherited the *ISink* properties. After the attachment of all sources to its dedicated sinks the whole transmission path is complete. A simple call of the method *getData* processes the incoming data and returns it to *mix*. After the HRTF rendering the original audio processing of *Mumble* is (re-)used. The use of the model of a virtual transmission path in a HRTF rendering approach offers a flexible and transparent way to insert new filters and HRTF datasets. With our approach it would be also possible to build a system with centralized audio rendering.

Due to the fact that each individual has its own HRTFs, our implementation of HRTF sound synthesis allows to quickly add new HRTF datasets with respect to the user. Our system works with datasets that are in the format of the well-known MIT HRTF database [9]. The user only has to select the new dataset through the provided graphical interface, and *Mumble* automatically reinitializes the filter structure with the new data.

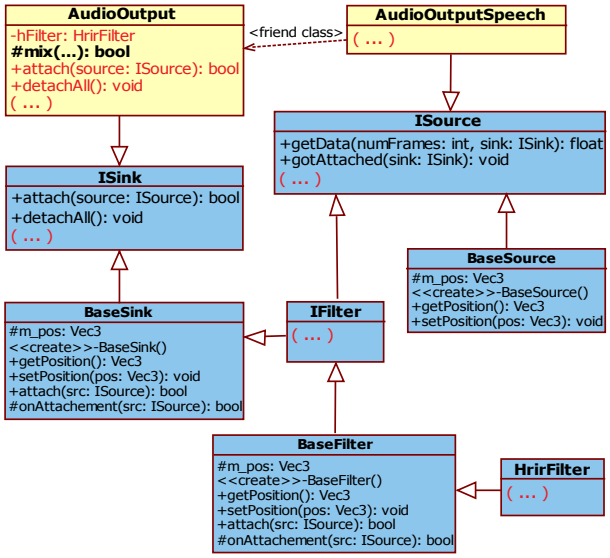

Fig. 6. Simplified *AudioOutput* class diagram

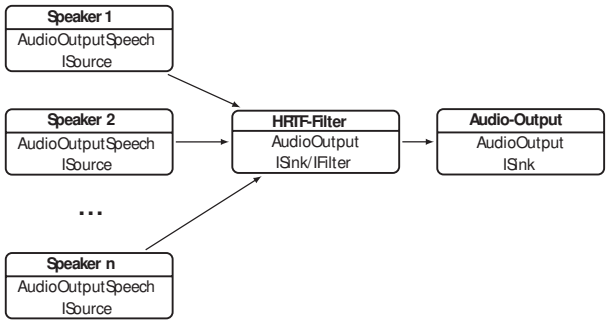

Fig. 7. HRTF Filter Model

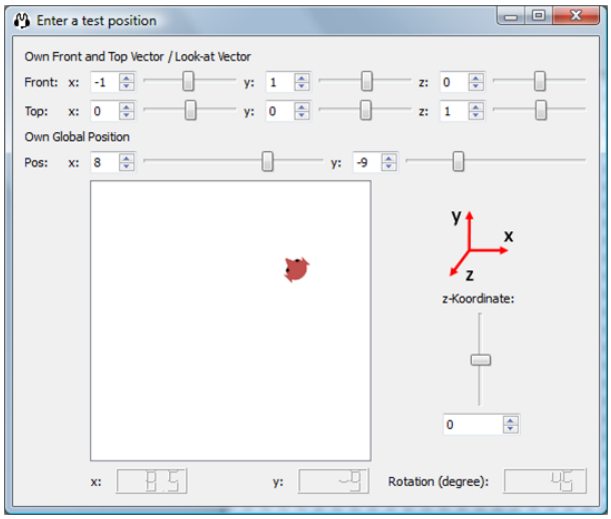

Fig. 8. Plugin to enter the own absolute position and head rotation

V. Results

In this section we give a short description of our 3D HRTF sound synthesis extension.

Mumble offers an interface for plugins to extract positions of conference or online game partners. In Figure 8 our plugin, which allows to adjust the user's global position within the virtual conference room as well as the head rotation by moving the sliders, is shown.

Figure 9 shows *Mumble's* main window with four active speakers. The position of the user and the positions of the other participants are indicated in the position window on the left side.

Depending on the relative position of conference participants, their mono signal is transfered to 3D audio by the corresponding HRTF. In this way, each participant's audio signal has also an unique directional information, enabling the listener to make use of the cocktail party effect. The cocktail party effect describes the ability of humans to concentrate on a conversation partner, even in situations, where lots of people are talking simultaneously [2]. Additionally, the loudness is modeled according to the distance between the participants, allowing two conference members to move away from the others to hold a private discussion. The user is completely free in choosing the spatial configuration of the different speakers.

Therefore, it is possible to use an optimal configuration for speech intelligibility like the one proposed in [10].

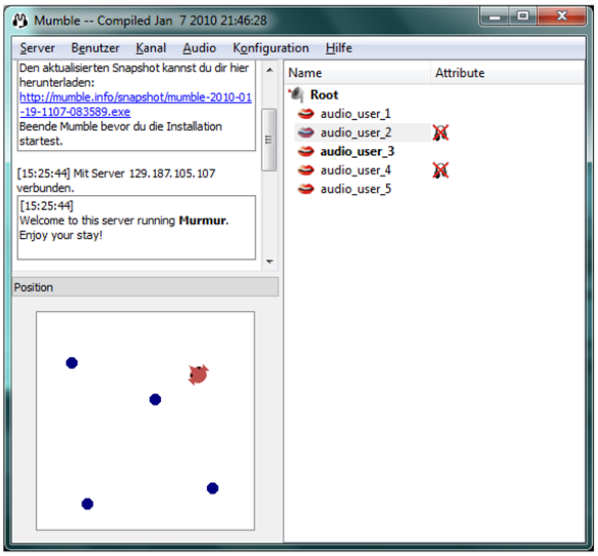

Fig. 9. Main window

There are several studies showing a better intelligibility of HRTF synthesized speech signals compared to mono and stereo signals [11]. Nevertheless, we conducted preliminary listening experiments to ensure the functionality of *Mumble* with 3D HRTF sound synthesis with low budget headsets in an office environment. Our first observations of the system supports the expectation that intelligibility of conferences using HRTF sound synthesis is increased, compared to mono and *Positional Audio* sound processing.

VI. Conclusion and Future Work

This paper describes an integration of HRTF sound synthesis into the conference software *Mumble*. Our 3D audio extension provides an improved identification of conference participants and increases intelligibility in situations with more than one active speaker (cocktail party effect).

With HRTF based sound synthesis, psychoacoustic effects are used to add additional information to the transmitted voice data. At the moment, we are working on publishing our HRTF-extension within the *Mumble* project. Moreover, it would be interesting to conduct further listening experiments to verify the results of our preliminary tests.

Acknowledgment

This work was fully supported by the German Research Foundation (DFG) within the collaborative research center SFB-453 “High-Fidelity Telepresence and Teleaction”.

References

[1] A. N. Mortlock, D. Machin, S. McConnell, and P. Sheppard, “Virtual conferencing,” *BT Technology Journal*, vol. 15, no. 4, pp. 120–129, 1997.

[2] B. Arons, “A review of the cocktail party effect,” *Journal of the American Voice I/O Society*, vol. 12, no. 7, pp. 35–50, 1992.

[3] M. Hyder, M. Haun, and C. Hoene, “Placing the participants of a spatial audio conference call,” in *IEEE Consumer Communications and Networking Conference-Multimedia Communication and Services, 2010*.

[4] J. Blauert, “An introduction to binaural technology,” in *Binaural and Spatial Hearing*, R. Gilkey, T. Anderson, Eds., Lawrence Erlbaum, Hilldale, NJ, USA, 1997, pp. 593–609.

[5] H. Møller, M. Sørensen, D. Hammershøi, and C. Jensen, “Head-related transfer functions of human subjects,” *Journal of the Audio Engineering Society*, vol. 43, no. 5, pp. 300–321, 1995.

[6] A. Torger and A. Farina, “Real-time partitioned convolution for Ambiophonics surround sound,” in *2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2001)*, 2001, pp. 21–24.

[7] W. Gardner, “Efficient convolution without input-output delay,” *Journal of the Audio Engineering Society*, vol. 43, no. 3, pp. 127–136, 1995.

[8] V. Pulkki and M. Karjalainen, “Localization of amplitude-panned virtual sources I: Stereophonic panning,” *Journal of the Audio Engineering Society*, vol. 49, no. 9, pp. 739–752, 2001.

[9] W. Gardner and K. Martin, “HRTF measurements of a KEMAR,” *The Journal of the Acoustical Society of America*, vol. 97, p. 3907, 1995.

[10] D. Brungart and B. Simpson, “Optimizing the spatial configuration of a seven-talker speech display,” *ACM Transactions on Applied Perception (TAP)*, vol. 2, no. 4, p. 436, 2005.

[11] R. Drullman and A. Bronkhorst, “Multichannel speech intelligibility and talker recognition using monaural, binaural, and three-dimensional auditory presentation,” *The Journal of the Acoustical Society of America*, vol. 107, pp. 2224–2235, 2000.