A Structural Parametric Binaural 3D Sound Implementation Using Open Hardware

Bruno Dal Bó Silva* and Marcelo Götz

Electrical Engineering Department
Federal University of Rio Grande do Sul - UFRGS, Porto Alegre, Brazil
bruno.silva@ims.ind.br, mgoetz@ece.ufrgs.br

Abstract. Most binaural 3D sound implementations use large databases with pre-recorded transfer functions, which are mostly prohibitive for real time embedded applications. This article focus on a parametric approach proposal, opening space for customizations and to add new processing blocks easily. In this work we show the feasibility of a parametric binaural architecture for dynamic sound localization in embedded platforms for mobile applications, ranging from multimedia and entertainment to hearing aid for individuals with visual disabilities. The complete solution, ranging from algorithms analysis and suiting, development using the Beagleboard platform for prototyping, and performance benchmarks, are presented.

Keywords: Binaural, Parametric, Beagleboard, Embedded, Digital Signal Processing

1 Introduction

Virtual sound localization is an interesting feature for innumerous types of applications, ranging from multimedia and entertainment to hearing aid for individuals with visual disabilities. Binaural sound localization is a technique that uses single-channel input signal, and by proper manipulation, can produce two different signals (left and right) channels. By these means a human user might perceive the source located on a determined point in space.

Head-Related Transfer Function (HRTF), which is usually well known technique employed for binaural sound localization, is a hard problem and computational intensive task. Such systems, enabling a dynamic changing of azimuth angle, usually requires the usage of dedicated computing hardware (e.g.[1–3]). Nowadays execution platforms found in mobile devices usually offers a notable computation capacity for audio processing, which is usually promoted by the inclusion of a Digital Signal Processing (DSP) processor. However, HRTF related algorithms must be well designed for such platforms to take advantage of the available processing capacity.

In our work we aggregate various methods and models of parametric 3D binaural sound for dynamic sound localization, propose some algorithm suiting

* Currently with IMS Power Quality

(specially for Interaural Level Difference - ILD filter) for embedded development, and validate them by prototyping using low-cost embedded hardware platform. In this work we present our first step towards a full-featured 3D binaural system, where we have implemented parametric DSP blocks on real-world hardware for feasibility evaluation.

The article is organized as follows. Firstly, in Sect. 2, we will introduce previous works on the field, followed by the models of our choice for embedded development and some considerations in Sect. 3. Then, the platform itself, technologic aspects and the employed algorithms adaptations are explained in Sect. 4. The experiment set-up including performance and localization quality results are shown in Sect. 5. Finally, Sect. 6 gives the summary, conclusions, and points out future work.

2 Related Work

Many previous researchers have invested on binaural sound localization. Two main areas of interest stand out: interpreting a dual-channel audio input so to discover the approximate position of the sound source (e.g. [4, 5]); and manipulation of a given single-channel input signal so to put it on a determined point in space as perceived by a human user. These two areas can be seen as the inverse of each-other, where the input of one is generally the output for the other. On this article we will explore the second field of research, knowing that much can be learned from the common physical principles involved.

Considering the problem of localizing a mono sound source for the human ear, we can still divide it in two great fronts: model-based and empiric localization. Many authors [1–3] rely on pre-recorded impulse-response databases and implement complex interpolations and predictive algorithms, varying from DSP-chip based algorithms to full-fledged System-on-Chip (SoC) solutions. Hyung Jung et al. [1] show a successful implementation of 5.1 surround sound system using hand-chosen responses, which works perfectly considering only 5 possible audio sources in space.

For more possible audio sources in a virtual space, almost infinite positions theoretically, we must have a rather large database of known responses and very complex interpolation algorithms, above all when considering as well the effects of elevation. Most of these systems rely on a public domain database of high-spatial-resolution HRTF [6]. Using such a database, implies storing it into the target execution platform.

We have relied on the second approach: a parametric model. For this we assume it is possible to approximate the physical response of the medium the sound is travelling in, the time and frequency response involved in the human shape (mainly the head and ears). We have searched successful models of such physical concepts, starting with [7], where Rayleigh gives us the prime of Psychoacoustics. Most of our model was inspired by the various works of Algazi, Brown and Duda [8–13] ,who have much contributed to the field. Their models use simplistic filters perfectly suitable for low-cost and satisfying results. For pinna-related

effects we have relied on the responses found in [14] and the spherical-model approximations given by [15].

3 Fundamentals

Most binaural 3D sound systems rely on a simple concept: the HRTF - Head Related Transfer Funcion. This macro-block represents the shaping suffered by the sound as it travels from source to the receiver's eardrums. The HRTF is usually divided in: ILD, ITD and PRTF, Interaural Level Difference, Interaural Time Difference and Pinna-Related Transfer Function respectively. We will disregard in this model the shoulder and torso effects, since they contribute mostly to elevation perception and our focus will be on azimuth. Fig. 1 shows the system's full conceptual block diagram.

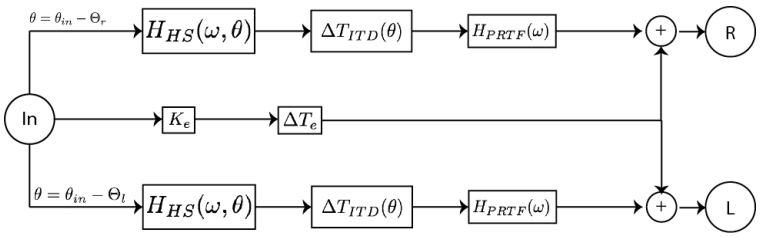

Fig. 1. Full localization model

3.1 ITD - Interaural Time Difference

ITD is conveyed by the difference in length of the paths to the left and right ears. This translates usually in a slight phase shift that is only perceivable for very low frequencies. By modelling the head as a sphere and considering the elevation angle ϕ to be always zero, that is, the sound source is always in the same plane as the listener's ears, we have the following model from [15]:

$$DL = \begin{cases} DLD & , DLD < L \\ L + DLA & , DLD \geq L \end{cases} \qquad DR = \begin{cases} DRD & , DRD < L \\ L + DRA & , DRD \geq L \end{cases} \tag{1}$$

where $\Delta D = DL - DR$ is the path difference, related to the ITD through the sound speed c. $ITD = \frac{\Delta D}{c}$. The geometric parameters are shown in Fig. 2.

The total ITD will be, then, discrete and taken in samples, according to the sampling frequency F_s. Both the linear and arc distances are solved using simple trigonometry and depend mostly on one parameter: the radius of the head, a_h.

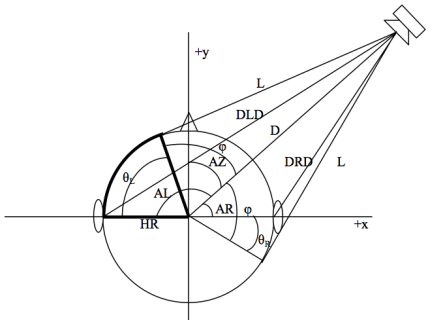

Fig. 2. ITD input parameters.

3.2 ILD - Interaural Level Difference

ILD is given by the Head-Shadowing effect, studied and modelled by Rayleight [7] as the solution of the Helmoltz equation for a spherical head. Brown shows an approximate model by a minimum phase filter in [14], also found in [16]:

$$H_{HS}(\omega, \theta) = \frac{1 + j\frac{\alpha\omega}{2\omega_0}}{1 + j\frac{\omega}{2\omega_0}} \tag{2}$$

where H_{HS} is the filter's frequency response, dependent on input azimuth angle θ, and α is given by:

$$\alpha(\theta) = \left(1 + \frac{\alpha_{min}}{2}\right) + \left(1 - \frac{\alpha_{min}}{2}\right) \cdot \cos\left(\pi\frac{\theta}{\theta_0}\right) \tag{3}$$

The parameter θ_0 fixes the minimum gain angle (α_{min}). Brown suggest that θ_0 should be 150°, for maximum match with Rayleigh's model, but we've found that it creates discontinuity in the frequency spectrum that is perceived as a fast "warp" of the sound source, so we've preferred to make $\theta_0 = 180°$ without major losses to the original model. Then, by simple variable mapping we have the filter responses for the left and right ears defined by:

$$H^l_{HS}(\omega, \theta) = H_{HS}(\omega, \theta - \Theta_l) \tag{4}$$

$$H^r_{HS}(\omega, \theta) = H_{HS}(\omega, \theta - \Theta_r) \tag{5}$$

The ILD model result is shown in Fig. 3. We can see that it provides a valid approximation to Rayleigh's solution and at the same time allows a very simple digital filter implementation. Notice symmetrical responses to the ear's reference angle overlap.

3.3 PRTF - Pinna-Related Transfer Function

With the ILD and the ITD we have modelled the general sound waveshape arriving at the listener's ears. The two previous blocks cover most of the physical

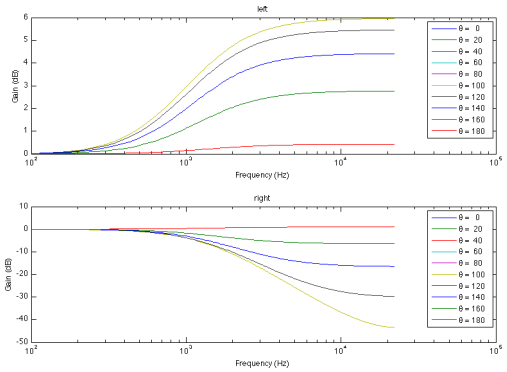

Fig. 3. ILD model result for left and right ears with $\theta = \{0, 180\}$

shaping, while the PRTF transforms the input wave even further, giving a more natural feeling to the user. We have used the models suggested in [14], which approximate the Pinna as a series of constructive and destructive interferences represented by notch and resonance filters. According to Spagnol et al., Pinna effects give many spectral cues to perceive elevation, we have found that adding their filter the azimuth plane ($\phi = 0$) we could create a rather superior localization feeling.

Spangol's general filter solutions are shown in Eq. 6 and 7. Table 1 shows the coefficients derived from the mentioned study, pointing the reflection (*refl*) and resonance (*res*) points with their central frequencies (f_C) and gain (G). The reflection coefficients will generate local minima in the frequency spectrum, whereas resonance frequencies will generate maxima, as it can be seen in Fig. 4.

$$H_{res}(Z) = \frac{V_0(1-h)(1-Z^{-2})}{1+2dhZ^{-1}+(2h-1)Z^{-2}} \tag{6}$$

$$H_{refl}(Z) = \frac{1+(1+k)\frac{H_0}{2}+d(1-k)Z^{-1}+(-k-(1+k)\frac{H_0}{2})Z^{-2}}{1+d(1-k)Z^{-1}-kZ^{-2}} \tag{7}$$

Table 1. PRTF filter coefficients.

Singularity	**Type**	f_C [kHz]	G [dB]
1	*res*	4	5
2	*res*	12	5
3	*refl*	6	5
4	*refl*	9	5
5	*refl*	11	5

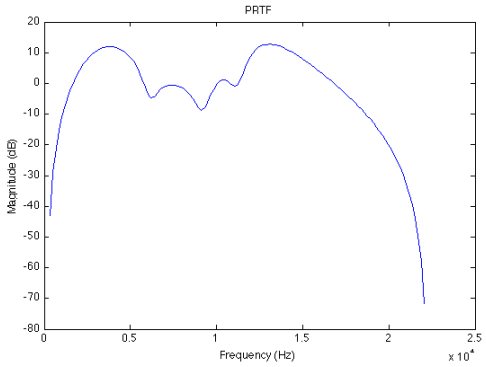

Fig. 4. PRTF filter result

3.4 RIR - Room Impulse Response

Considering the effects of the three previously mentioned models, we can achieve an acceptable result. But it suffers from lateralization, where the listener sometimes perceives the sound as coming from inside the head. The digitally processed waveshape does not have, yet, the necessary cues for proper localization. Brown suggests that a very simple RIR can make a difference [13]. As we can see in Fig. 1, there is a direct signal path that is not seen as dependent on the input angle θ. That is a simple echo. By simply adding to the resulting binaural signals the original mono input with delay and attenuation we can reduce lateralization by a considerable amount, increasing externalization. We have used a 15ms delay with 15dB attenuation.

3.5 System Response

The system's final response is shown in Fig. 5. The visual representation uses frequency as the radius, azimuth as argument and the system's output is color-coded in the z-axis. The polar plot clearly shows the different energy regions dependant on the azimuth and the reference angle – left or right ear. The distinctive "waves" are due to the Pinna effects, whereas the global energy difference is given by the Interaural Level Difference.

4 Development

Our main goal was to port the mentioned filters and models to an embedded platform. We have chosen the Beagleboard [17], a community-supported open hardware platform which uses Texas Instruments' DM3730, containing one ARM Cortex-A8 and one DSP C64x. Being derived from the OMAP platforms, this chipset allows real-world performance validation using embedded Linux relying on the two available cores: a General Purpose Processor (GPP) and a DSP

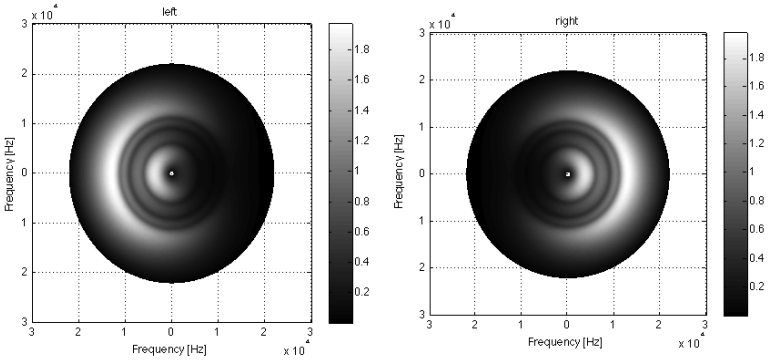

Fig. 5. System's final response in polar plot.

Processor. We have loaded the Beagleboard with a custom lightweight Angstrom distribution built using Narcissus [18]. Narcissus also generates a full-featured development environment ready for cross-compiling.

The building environment has three standing points: `gcc-arm` generates an ARM-compatible linux ELF executable; `ti-cgt` generates DSP-compatible functions, bundled into a library file; then we call an external tool available for easy co-processing on this hybrid chips – C6RunLib. This tool, supported by Texas Instruments, relies on a kernel module called *dsplink* that handles communication between processors. C6RunLib acts as a linker, putting together the ARM code with the DSP library and adding necessary code for remote procedure calls. When a function that is run on the DSP side is called from within ARM code, *dsplink* is called and uses shared memory to actually have the function operated by the other core. We have not entered the details of C6RunLib's inner workings, but one important feature was missing: asynchronous cross-processor function calls were not available – we have solved this issue by letting the ARM-side thread hang while the DSP is busy operating the entry-point function.

The program consists of three main parts: User Interface, Audio Output and Signal Processing. The latter is done on the DSP core, while the others are performed by the GPP (ARM Processor). The User Interface allows the user to dynamically control the azimuth angle θ, azimuth automatic change rate, output volume, turn off the filters individually (so to benchmark localization quality easily without the need to rebuild the solution) and chose among three possible infinite loop audio inputs: noise, an A tone (440Hz) or a short phrase from Suzanne Vega's Tom's Diner (benchmark track suggested in [16] for its pure-voice contents). Audio Output and Signal Processing blocks require real time operation to operate smoothly and are detailed on Sect. 4.2.

4.1 Algorithm Suiting

The four filter blocks were implemented using digital filters with some help from TI's DSPLIB library[19] for better performance. Algorithm suiting for real time

digital processing was the first step to take. The delay filters (ITD and RIR) are simple circular buffer filters, only the ITD has a variable delay given by the azimuth. PRTF is modelled by a fixed-point 16-tap truncated FIR filter calculated from Eq. 6 and 7 and using the coefficients from Table 1. Since PRTF is not sensible to azimuth changes, it is actually processed before we split the signal into two separate outputs, which is possible because all the filters are linear and the azimuth change is considerably slower than the system's response.

ILD algorithm is more complex because the filter's coefficients change with the azimuth. The minimum phase filter presented on Eq. 2 when sampled becomes:

$$H_{HS}(Z,\theta) = \frac{\left(\frac{\alpha(\theta)+\mu}{1+\mu}\right) + \left(\frac{\mu-\alpha(\theta)}{1+\mu}\right) Z^{-1}}{1 + \left(\frac{\mu-1}{1+\mu}\right) Z^{-1}} \tag{8}$$

where $\mu = \frac{\omega_0}{F_s} = \frac{c}{a_h \cdot F_s}$. By solving Eq. 8 symbolically on the time domain, being $h[n]$ the related impulse response at sample n we have the following generalization:

$$h[n] = \begin{cases} \frac{\alpha+\mu}{\mu+1} & , n = 0 \\ -\frac{2\mu(\alpha-1)}{(\mu+1)^2} & , n = 1 \\ h[n-1] \cdot \frac{(1-\mu)}{(\mu+1)} & , n > 1 \end{cases} \tag{9}$$

Using Eq. 9 the ILD is a time-variant fixed-point 16-tap FIR which is recalculated for every input buffer (given the azimuth has changed). On top of the presented model we have also added a low-pass filter to the input azimuth to avoid angle "leaps".

4.2 Prototyping System

As discussed in Sect. 4, we have to build blocks that rely on real time operation for our purposes. For audio output we have used Jack Audio Connection Kit [20], an open library made for this kind of processing. It interfaces with Advance Linux Sound Architecture (ALSA) and tries to guarantee audio transmission without over or underruns. Jack is called by instantiating a deamon that waits for a client to connect to its inputs and outputs. The client registers a callback function that has access to the deamon's buffers. In our application we have used 512 frames-long buffers at a 48kHz sampling frequency, meaning each buffer is valid for around 10ms, which is our processing deadline and output latency. We have not used Jack as the input source, instead the audio input is read from a file and kept in memory. To simulate real time audio capture, this memory area is used to feed new input every time the callback interrupt is called for new output, thus validating the real time premise.

The Jack callback function then signals another thread that sends the input signal to the DSP and receives back (always in a shared memory area) the processed response. All 512-samples buffers are signed with an incremental integer and constantly checked for skipped buffers, thus providing certainty that every input is processed and that every output is played back. Along with the signal

buffers, the main application sends a structure containing the controllable parameters discussed previously. Fig. 6 illustrates the system's building blocks and their mutual operation. The left and right trunks operate both at application level (running on GPP) and are ruled by posix semaphores. The middle trunk operates inside Jack's callback function, which is triggered by the Jack daemon once the client is properly installed. The rightmost branch is the C6RunLib function that is processed on the DSP side, remembering that the Sync thread stays blocked during DSP operation.

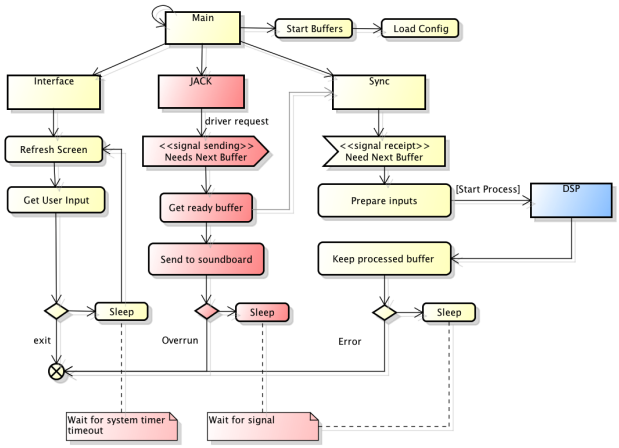

Fig. 6. System thread schema

Fig. 7 shows the function which is operated on the DSP, it follows almost exactly the flowchart presented previously. When done filtering the input signal, the function will sign the output buffers and return, letting the caller thread unblock and renders the newly processed buffers available for the next audio output callback. We have found that the very first call to a function running on the DSP side takes considerably longer than the subsequent ones because of the time needed to load the application code to the other processor. This task is masked by the facilities provided by C6RunLib but must be taken into account, otherwise the system will always shut down on the first buffer because of this latency and the fact that the callback function kills the program if an underrun is detected.

5 Experimental Results

We have tested the system's localization quality and perceived accuracy with a set of user trials. Three main set-ups were tested. One in which another person controls the azimuth freely and the subject must grade the localization feeling

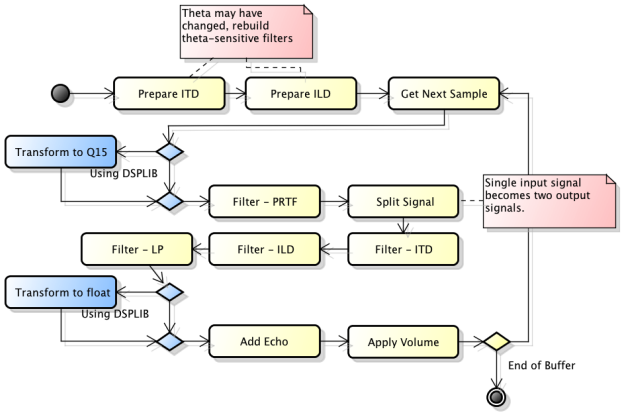

Fig. 7. Signal Processing Function

and try to point out where the sound is coming from. On another test subjects were free to change azimuth themselves. The grades being "Good", "Average" and "Poor". On a last test users were asked to point out where the sound source was in virtual space. The accuracy test is divided in two parts: guessing the azimuth angle of a "warping" sound source that instantly goes from one point to another and of a "moving" source which moves slowly until rest. In the accuracy test we have gathered mean and maximum perceived azimuth errors ϵ_μ and ϵ_M. All users were given the same test points set in random order.

We have noticed that most subjects feel a stronger localization when they have control over the azimuth, complying with the fact that a person with no sensing disabilities will rely on a sum of those to pinpoint an object on space, usually failing when depending solely on sound – create somewhat a placebo effect. When trying to point where the sound source is in a virtual space, users will always get the right quadrant if the sound source is "warping", but will generally fail to pinpoint angle giving $\epsilon_\mu = 43°$, $\epsilon_M = 70°$. When the subject is able to hear the virtual source moving to a resting position accuracy increases to as much as about 30 degrees without mistakes ($\epsilon_\mu = 21°$, $\epsilon_M = 30°$), totalling 12 clear zones in the azimuth plane.

The quality tests were also repeated three times for each of the available input samples. It was observed that the pure tone does not have enough spectral cues for localization and generates confusion most of the time. Noise is next on the list, but, for its high frequency components, it is still not as clear as it could be since the complex frequency response of the implemented system is on the lower side of the spectrum.

The mentioned accuracy results are valid for the third sample: human voice. Because of the short bandwidth and, some authors will argue, an inherent capability to localize voice this third input sample grades enormously better than the previous ones – which is largely consistent with the mathematical model. Lastly, fewer tests were run turning some filtering stages on and off so to observe the

real necessity of each one. It was seen that the ITD and ILD work very closely together (although the ITD plays a void role with noise input, since phase cue is basically non-existent because of high frequencies). PRTF helps in fixing the sound source to the azimuth plane, the system without it was described as "noisy and confusing" by subjects, inducing that the PRTF wraps the spectrum to a more natural balance. Some subjects experienced lateralization when the RIR was turned off, losing completely the source in the virtual space.

As for performance, the system was built considering the fact it would not miss deadlines, so the described implementation was validated. We have tested some other parameters to test the system working under stress. It was observed that reducing the frame size to 256 samples (5ms deadline) would cause overruns depending on the operating system's load. For example, the system would run perfectly using its own video board and an attached USB keyboard, but would fail through an SSH connection. Although the same number of samples is processed per second, that is the sampling frequency remains unchanged, the cross-processor calls create noticeable overhead – a problem that could be approached by getting into the inner workings of C6RunLib.

Also, we have made it possible for the program to run fully on the GPP side, letting the DSP completely unused, and compared the CPU load reported by the operating system. When the filters are run by the second processor, the main CPU stays practically idle reporting in average 3% load. When running the complete system on the GPP side, the CPU goes almost to maximum load, floating most of the time around 97% . We were not able to sample the DSP load because the benchmarking functions provided by the libraries would crash the system. To give an approximation of DSP load, we have made the processing functions run redundantly n times, reaching overall failure at $n = 3$, so we estimate the DSP core load between 30% and 50% of its full capacity. The load tests were performed with a constantly changing azimuth, so the filter parameters would be constantly changing and we could observe the worst case scenario.

6 Conclusions

In this work we propose a full-featured 3D binaural system for sound localization. We successfully aggregated various methods and models of parametric 3D binaural, creating parametric DSP blocks. Furthermore, we have shown the feasibility by implementing and evaluating these blocks on a low-cost embedded platform. In our tests the azimuth angle was dynamically changed and clearly perceived by the user with enough accuracy, specially for human voice sound source.

Depending on the operating systems's load, some buffers overrun occurs, which decreases the output quality. This is caused probably by cross-processors calls, which relies on C6RunLib library. So, next steps in our work will be the C6RunLib internals analysis and propose some implementation solutions for this problem. We will also improve the model to consider elevation and more complex room responses.

References

1. Kim, H.J., Jee, D.G., Park, M.H., Yoon, B.S., Choi, S.I.: The real-time implementation of 3D sound system using DSP. In: Vehicular Technology Conference, 2004. VTC2004-Fall. IEEE 60th. Volume 7. (Sept. 2004) 4798 – 4800
2. Fohl, W., Reichardt, J., Kuhr, J.: A System-On-Chip Platform for HRTF-Based Realtime Spatial Audio Rendering With an Improved Realtime Filter Interpolation. International Journal on Advances in Intelligent Systems **4**(3 & 4) (2011) 309 – 317
3. Sakamoto, N., Kobayashi, W., Onoye, T., Shirakawa, I.: DSP Implementation of Low Computational 3D Sound Localization Algorithm. In: Signal Processing Systems, 2001 IEEE Workshop on. (2001) 109 –116
4. Raspaud, M., Viste, H., Evangelista, G.: Binaural Source Localization by Joint Estimation of ILD and ITD. Audio, Speech, and Language Processing, IEEE Transactions on **18**(1) (jan 2010) 68 – 77
5. Rodemann, T., Heckmann, M., Joublin, F., Goerick, C., Scholling, B.: Real-time Sound Localization With a Binaural Head-system Using a Biologically-inspired Cue-triple Mapping. In: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on. (Oct. 2006) 860 – 865
6. Algazi, V.R., Duda, R.O., Thompson, D.M., Avendano, C.: The CIPIC HRTF Database. In: Proc. 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2001), New Paltz, NY, USA (October 2001)
7. Rayleigh, J.: The theory of sound. Number v. 1 in The Theory of Sound. Macmillan (1894)
8. Algazi, V.R., Dalton, Robert J., J., Duda, R.O., Thompson, D.M.: Motion-Tracked Binaural Sound for Personal Music Players. In: Audio Engineering Society Convention 119. (10 2005)
9. Algazi, V.R., Duda, R.O., Thompson, D.M., Avendano, C.: The Cipic HRTF Database. CIPIC U.C. Davis (2001)
10. Avendano, C., Algazi, V.R., Duda, R.O.: A Head-and-Torso Model for Low-Frequency Binaural Elevation Effects. IEEE Workshop on APplications of Signal Processing to Audio and Accoustics (1999)
11. Algazi, V.R., Avendano, C., Duda, R.O.: Estimation of a Spherical-Head Model from Anthropometry. National Science Foundation (2001)
12. Brown, C.P., Duda, R.O.: An Efficient HRTF Model For 3-D Sound. Master's thesis, University Of Mariyland, San Jose State University (1997)
13. Brown, C.P., Duda, R.O.: A Sructural Model for Binaural Sound Synthesis. IEE Transactions on Speech and Audio Processing **6** (1988)
14. Spagnol, S., Geronazzo, M., Avanzini, F.: Structural modeling of pinna-related transfer functions. In: In Proc. Int. Conf. on Sound and Music Computing. (2010)
15. Miller, J.D.: Modeling Interneural Time Difference Assuming a Spherical Head. Master's thesis, Stanford University (2001)
16. Zölzer, U., Amatriain, X.: DAFX: digital audio effects. Wiley (2002)
17. GolinHarris: Beagleboard.org. `http://beagleboard.org` (2011)
18. Narcissus: Narcissus Angstrom Distribution. `http://narcissus.angstrom-distribution.org/`
19. TI: DSPLIB. `http://processors.wiki.ti.com/index.php/DSPLIB`
20. Davis, P.: Jack Audio Connection Kit. `http://jackaudio.org/`