3D SOUND IMAGING WITH HEAD TRACKING

William Fitzpatrick, Mark Wickert, and Sudhanshu Semwal
College of Engineering and Applied Science, University of Colorado, Colorado Springs
CO 80933-7150, Email: wfitzpat@uccs.edu, mwickert@uccs.edu, ssemwal@uccs.edu

ABSTRACT

In Virtual Reality (VR) applications, an important measure of quality is the extent to which the user believes he is a part of (or immersed in) the virtual world which is being simulated. In this paper our focus is sound that is emanating from a source positioned in a particular 3D location in the real world. The ability to determine the 3D position of a sound source is called sound localization. A key element in localization is the Head Related Transfer Function (HTRF) and the associated impulse response. Here we use the U.C. Davis HTRF database along with a real-time DSP engine to implement the localization filters. By detecting the pose of the listener’s head, the apparent position of a sound source relative to the listener is adjusted to accommodate the movement of the listener’s head, thereby making the sound source appear to be at a fixed position in 3D space.

Index Terms— sound localization, head related transfer function, head tracking

1. INTRODUCTION

In Virtual Reality (VR) applications, an important measure of quality is the extent to which the user believes he is a part of (or immersed in) the virtual world which is being simulated. VR applications can be enhanced by including human senses other than vision. These other senses may include touch (“haptics”), smell and sound. In this project a method is developed which can be used to create the illusion that sound is emanating from a source positioned in a particular 3D location in the real world.

2. SOUND LOCALIZATION

The ability to determine the 3D position of a sound source is called sound localization. In humans, two primary cues are used to localize the source of a sound.

2.1. Interaural Level Difference (ILD)

The first cue is the relative loudness of a sound as heard in one ear compared to the other. This is called the Interaural Level Difference, or ILD. The reasons that a sound should be louder in one ear than the other are twofold: First, the ear that is closest to the source of sound would hear it louder simply by virtue of being closer, since the sound level (pressure) diminishes in inverse proportion to the distance to the source [1]. However, for distant sounds the difference in sound level heard by each ear becomes negligible. More importantly, the head has a “shadowing effect” – the head partially blocks propagation of the sound to the ear on the opposite side of the head. This effect is more pronounced for high frequencies than for low, because the sound waves for low frequencies have a wavelength much larger than the head dimensions and tend to diffract around the head and reach the opposite ear.

2.2. Interaural Time Difference (ITD)

The other primary cue is the time delay difference with which the sound reaches one ear compared to the other, called the Interaural Time Difference, or ITD. For frequencies below 800 Hz, the time delay from one ear to the other is no more than ½ wavelength of the sound, so it is possible for the brain to evaluate the phase difference for these low frequencies. For frequencies above 1600 Hz, it is no longer possible to use phase (because there is multiple cycles with the same phase difference), and the brain evaluates group delays instead. This can be thought of as determining the relative delay of the onset of a sound or an amplitude-related feature in the sound envelope. For intermediate frequencies between 800 and 1600 Hz, a combination of the two effects is used.

2.3. Cone of Confusion

If these two mechanisms were the only ones used to judge the position of a sound source, it would not be possible to do so unambiguously, because at any given combination of ILD and ITD, the only determination about the location of the sound source which can be made is that the locus lies somewhere on the surface of an imaginary cone, called the cone of confusion. This is illustrated in Figure 1.

2.4. The Head Related Transfer Function (HRTF)

Fortunately, the human hearing mechanism has another feature which helps to further pinpoint the location of the
sound source. The ridges, creases and other structures of the external ear (the pinna) are radially asymmetric, and serve to filter sounds coming from different directions in such a way that the filtering is unique to each direction.

![Figure 1. Cone of confusion](image)

To a lesser degree, the head and torso also affect the spectral content, especially for the ear on the opposite side of the head from the sound source. Together, these features (pinnae, head and torso) serve to color the spectrum of sound in a way that is unique to each possible position of a sound source in 3D space. The brain is able to analyze the spectral coloration to help resolve the ambiguity inherent in the cone of confusion. This spectral coloration can be measured experimentally, and together with the ILD and ITD comprise what is known as the Head Related Transfer Function, or HRTF. In order to create the illusion that a sound source is located at any desired position in 3D space, this sound signal can be processed with an HRTF which is a reasonable match for a given listener.

### 2.5. Head movement

In order to fine-tune the location of a sound source, many animals (including humans) often turn their head in an effort to resolve any remaining ambiguities. Because of this, the believability of a 3D sound simulation can be enhanced by properly accounting for head movement. Conversely, the illusion can be easily shattered if head movement is not taken into account.

### 3. THE CIPIC HRTF DATABASE

In practice, the HRTF is measured indirectly by measuring the Head Related Impulse Response, or HRIR. The HRIRs for numerous test subjects have been measured and the results made freely available on the web. For this project, the CIPIC HRTF Database, compiled at U.C. Davis [3] was selected.

The CIPIC database consists of HRIRs sampled at 48 kHz for each of 25 different azimuth and 50 elevation combinations (1250 HRIRs per ear per record, spaced roughly 5° apart).

The database is organized as one file per subject. Each HRIR file (hrir_final.mat) is in a MATLAB .mat format, which is a binary format that supports matrices and complex data structures. Embedded in each .mat file is a pair of 3-dimensional double-precision arrays, hrir_l and hrir_r, representing the Head Related Impulse Responses for the left and right ears, respectively. The dimensions of each array are: 25 (azimuth values) by 50 (elevation values) by 200 (number of samples in the impulse response).

### 4. THE HRTF IMPLEMENTED AS AN FIR FILTER

To process a sampled audio signal with the HRTF, one needs only to implement an FIR which convolves the audio signal with the HRIR. For an HRIR length of 200, a 200-tap FIR is needed for each channel which implements the following difference equation:

\[ y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] \ldots, \]

\[ n = 0, 1, 2, \ldots, 199 \]

In this equation, the \( y[n] \)'s are the output samples, the \( x[n] \)'s are the input samples and the HRIR sequence is used for the \( b_i \)'s (filter coefficients).

### 5. PROJECT DESCRIPTION

This project is comprised of two major components: real-time audio signal processing and head tracking.

#### 5.1. Real-Time Audio Signal Processing

##### 5.1.1. Hardware development platform

For this project, the real-time audio signal processing is performed using dedicated Digital Signal Processing (DSP) hardware. Computation of the HRTF is very processor-intensive, and DSPs have hundreds of cores and are well-suited for this type of processing. For this project, I selected the Texas Instruments OMAP-L138, a combination of an ARM9 processor and a C6748 floating point DSP processor. To enable software development, I used the Zoom™ OMAP-L138 Evaluation Module (EVM), developed jointly by Texas Instruments and LogicPD to support OMAP-L138 development. The hardware was supplied thanks to [4].

##### 5.1.2. DSP software development

To design the FIR, DSP code development was accomplished using Code Composer™ Studio v4.2.5, an Eclipse-based IDE developed by TI.

Normally, DSP-only development for this DSK is made unnecessarily complex by the necessity of managing program loading, initialization, etc. Fortunately, the book, Real-Time Digital Signal Processing from MATLAB to C with the TMS320C6x DSPs [5] provides a number of utilities, sample programs and support libraries to simplify
the process. The winDSK8 program allows students to perform DSP experiments without the need to learn all board setup and programming details. To use it, the board is connected to a PC via the RS232 serial debug. Prior to using winDSK8 the first time, the winDSK8 kernel must be programmed into the DSK’s flash memory via the RS232 port. The winDSK8 Windows host program can then be run on the PC, which communicates with the DSK via the serial port. Using it, various DSP functions can be implemented and real-time processing performed on an audio signal using the onboard C6748 floating point DSP processor. The audio signals are accessed using the line-in and line-out 3.5mm audio jacks.

In order for the winDSK8 host program to transfer data to and from the DSK, the host program must know the addresses of the variables stored on the DSP. A dynamic link library file (C8X CONTROL DLLxx.DLL) contains a function that will return a variable’s address, provided the variable name has been stored in the executable file’s symbol table.

5.1.3. Host software development

In addition to transferring data to the DSP, another function of the host program is to load and run the precompiled DSP code on the DSK. For this project, we didn’t use the winDSK8 DSP or host programs. Within the winDSK8 package was another sample program which provides a simple gain slider control on the host program, used with a simple gain control DSP program. We used this program instead, using it as a skeletal base for development of my HRTF DSP design using Code Composer Studio (in C code) and Microsoft Visual Studio 2010 for development of the host software (in C++).

5.1.4. Host-DSK system

In order to make use of the CIPIC HRTF database, we needed to design a 200th order FIR for both the left and right ears, since the length of the impulse response in the database is 200 samples (per azimuth per elevation per ear). We decided to have the DSP responsible only for DSP processing, leaving the other tasks to the host program, including transferring the FIR coefficients to the DSK over RS232 every time the sound position changes. The data contained in the HRTF database are double-precision; however the C8X CONTROL functions contained in the library only operate on floating point numbers. For this reason, we recast the FIR coefficients as float. Other reasons for doing this: calculations on the DSP involving floating point numbers execute faster than double-precision, data transfer over the serial port takes twice as long for double-precision than it does for floating point. Also, floating point numbers provide more than enough precision for this application.

So, the concept on the DSP side is simple: perform the FIR calculations on the sound samples as they arrive, using whatever coefficient values happen to be stored in the coefficient’s memory address. Whenever new coefficient values arrive from the host program, update the values stored in this same coefficient memory space.

5.2. Head Tracking

The term head tracking refers to a process in which the position and orientation (the pose) of the user’s head relative to some fixed frame of reference is calculated and then used to adjust the way VR data is presented to the user.

For this project, head tracking functionality has been added to accommodate head movement. Without it, as soon as the listener moved his head, it becomes immediately apparent that the sound is moving with the headphones, not fixed in a physical 3D spatial position. Head tracking can be used to improve the believability of a simulated sound source in 3D. By detecting the pose of the listener’s head, the apparent position of a sound source relative to the listener can be adjusted to accommodate the movement of the listener’s head, thereby making the sound source appear to be at a fixed position in 3D space.

Also, head tracking minimizes the tendency for a sound imaged directly in front to be perceived as coming from inside the head. In addition, head tracking minimizes the effect of a mismatch between the HRTF subject file used and the user’s actual HRTF [6] [7].

In order to implement head tracking, we chose faceAPI, an SDK developed by Seeing Machines [8]. This API provides libraries and example projects that can be added to a C or C++ project. This API makes use of a webcam (or other camera) to track facial features, extracting the position and orientation of the user’s head relative to the camera with 6 degrees of freedom (Figure 2).

![Figure 2 faceAPI coordinate frames.](image)

Once the position and orientation of the user’s head relative to some fixed spatial location is determined, the angle and distance to a virtual sound source can be calculated, and the appropriate HRTF FIR filter parameters can then be selected. As the user moves his head, the FIR parameters are adjusted accordingly, giving the illusion that the sound source is located at some particular position in
space. The ultimate goal for this is for the user (wearing headphones) to be able to move around freely, while a sound source is simulated at some specified (real world) 3D location.

5.3. Parallax Correction

The CIPIC HRTF database was generated by using a sound source at a fixed distance from the subject’s head (1 meter). Attempts to simulate a sound source at some distance other than the reference distance by simply adjusting the sound volume won’t work – the sound doesn’t seem to move closer or farther away, it simply sounds louder or quieter. The problem is caused by parallax error. This is illustrated in Figure 3 for a sound source in the horizontal plane. If the sound source is at the reference distance of 1 meter (point A), the FIR coefficients can be taken directly from the database entry corresponding to an azimuth angle of 0°. However, if the source is located much closer at point B (0.3 meters), the FIR coefficients for each ear must be taken from different azimuth value entries, as illustrated.

![Figure 3 Sound parallax.](image3.png)

To determine which set of FIR coefficients to use, one must calculate where the “sound ray” going from the ear to the sound locus intersects with the reference sphere by solving the equation of the sphere simultaneously with the equation of the line. Also, we need to find out the proportional distance to the sound locus compared to the distance to the reference sphere so that the inverse-distance law can be applied to the sound amplitude. Both these things can be found in a straightforward manner if we use a parametric representation for the equation of the line. This is illustrated in Figure 4.

![Figure 4 Finding the intersection of the sound ray with the reference sphere using parametric equations.](image4.png)

The sound ray can be described in parametric form by three equations:

\[ x = x_0 + t(x_1 - x_0) = x_0 + t\Delta x \]

The parameter \( t \) represents the distance along the sound ray to the intersection with the reference sphere relative to the length of the ray. For values of \( t \leq 0 \), the sound ray intersects the sphere, and the value of \( t \) represents the inverse of the distance to the sound source relative to the reference sphere. For example, if \( t = 0.5 \), the intersection point is midway between the sound source and the ear. In other words, the sound source is twice as far from the ear as the distance to the reference sphere. Since the sound would seem half as loud as it would at the reference distance because of the inverse-distance law, \( t \) is equal to the inverse distance and can be used directly as a gain value used to scale the perceived sound level for any given distance. This result is extremely convenient since the parameter \( t \) is calculated anyway during the process of the solution of the intersection point, and no additional calculations for determining distance need be performed. By the way, for sound sources inside the reference sphere, the solution finds the intersection point of the extension of the sound ray. Since \( t > 1 \) in this case, the sound would be amplified as it should. As another example, if the sound source is halfway between the ear and the reference sphere, the extension of the sound ray to the reference sphere is twice the distance, \( t = 2 \), and the sound would be heard twice as loud.

Now we need to find the \((x, y, z)\) coordinates of the intersection of the sound ray (or the extension of it) with the reference sphere. If we stay with the units from the CIPIC database (metric), the radius of the reference sphere is 1
meter and has its center at (0, 0, 0); therefore the equation for the reference sphere is simply:

\[ x^2 + y^2 + z^2 = 1 \]

Substituting the parametric equivalents from above into this equation yields:

\[(x_0 + t\Delta x)^2 + (y_0 + t\Delta y)^2 + (z_0 + t\Delta z)^2 = 1 \]

Let \( r \) represent the distance from the ear to the center of the head (roughly 8 cm, on average). The sound ray origin is therefore \((x_0, y_0, z_0) = (r, 0, 0)\). Substituting these values into the above equation, expanding and grouping according to powers of \( t \) yields:

\[ [(x_1 - r)^2 + y_1^2 + z_1^2]t^2 + [2r(x_1 - r)]t + (r^2 - 1) = 0 \]

This is a quadratic equation in \( t \),

\[ at^2 + bt + c = 0 \]

where,

\[ a = (x_1 - r)^2 + y_1^2 + z_1^2 \]
\[ b = 2r(x_1 - r) \]
\[ c = r^2 - 1 \]

Solving for \( t \) using the quadratic formula yields:

\[ t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

which yields two values for \( t \). As long as \( r < 1 \) (extremely likely, since otherwise the head would have a radius greater than 1 meter!), the above equation always yields one positive value of \( t \) (representing the intersection of the sound ray with the reference sphere in the forward direction), and one negative value (the intersection of the extension of the sound ray in the reverse direction). At any rate, in order to pick the correct value, the calculation in the software must pick the positive (max) value for \( t \).

The coordinate values for the intersection point of the sound ray (or its extension) with the reference sphere can now be found by using the correct value of \( t \) from above in the original parametric equations, substituting the ear coordinates, \( P_0 = (x_0, y_0, z_0) = (r, 0, 0) \),

\[ x = x_0 + t(x_1 - x_0) = r + t(x_1 - r) \]
\[ y = y_0 + t(y_1 - y_0) = ty_1 \]
\[ z = z_0 + t(z_1 - z_0) = tz_1 \]

The Interaural Polar Coordinate System used by the CIPIC HRTF database is defined as follows (see Figure 5):

- The polar axis passes through the centers of the ears.
- The plane containing the polar axis and the sound source location is tilted by some angle \( \phi \) from the horizontal, the elevation angle. This angle varies from -90° (directly below), to 0° (straight ahead), to +90° (directly overhead), to +180° (directly behind), to +270° (directly below). Because making measurements is difficult for positions below the test subject, these angles are restricted to the range -45° to +230.625° in increments of 5.625°.
- The second coordinate, the azimuth angle (\( \theta \)) is given by the angle within the elevation plane measured from the midsagittal or vertical median plane to the vector to the sound source location. It varies from -90° (straight left), to 0° (straight ahead), to +90° (straight right) in increments of 5°. The increments become more widely spaced apart near +90°.
- The third coordinate (radius) is fixed at one meter for all database measurements.

In order to access the appropriate FIR coefficients from the HRTF database, the Cartesian coordinates of the sound ray intersection with the reference sphere obtained above must be transformed into the Interaural Polar coordinate system. This transformation is illustrated in Figure 5. The point \( P_1 = (x_1, y_1, z_1) \) is the sound locus. First the elevation angle is calculated as: \( \phi = \tan^{-1}\left(\frac{y_1}{z_1}\right) \). The c++ function: \( \text{atan2}(y_1, -z_1) \) properly protects against division by 0, and generates the correct angle in all four quadrants. Since the range of elevation values in the Interaural Polar coordinate system vary from \(-90° < \phi < 270°\), but the \( \text{atan2()} \) function returns values in the \(-180° < \phi < 180°\) range, the values in the range \(-180° < \phi < -90°\) need to be checked for and translated into the \(180° < \phi < 270°\) range by the software.

The azimuth angle is calculated simply as: \( \theta = \sin^{-1}x \), since the hypotenuse (which is the same as the radius of the reference sphere) is equal to 1.

Once the elevation and azimuth values are known, the values used to index into the HRTF database are selected by picking the closest angular value available in the database.

5.4 Putting it all Together

So far, we have the ability to simulate a sound at a certain position relative to the head by filtering the sound with the HRTF in real time using dedicated DSP hardware. The position of the sound can be made to appear at distances other than the 1 meter reference distance by using parallax correction techniques with parametric equations. We have a method for converting Cartesian coordinates to interaural polar coordinates and then using these to index into the HRTF database to retrieve the correct coefficient values. Finally we have a way to track the listener’s head position...
relative to a fixed point in space. What is still missing is a way to adjust the apparent position of the sound relative to the listener’s head to accommodate head movement. This is done with coordinate transformation algorithms.

The Windows control program, HRTF_HT.exe was built using Microsoft Visual Studio 2010 using C++, integrating elements from winDSK8 and faceAPI (see Figure 6). The GUI includes sliders for positioning the virtual sound source, a list box for selecting which subject’s HRTF data file is to be used and other controls. There is also a live video display window which shows the face of the user (with left/right reversal, like a mirror), overlaid with the facial landmarks which were detected by faceAPI (small blue dots). There is also a 3D “mask” (yellow lines) which represents the 3D position of the landmarks. The calculated position and orientation of the head coordinate frame (refer to Figure 2) is shown overlaid as the red arrow (x axis), green arrow (y axis) and blue arrow (z axis, positive direction going toward the back of the head). Performance data is shown overlaid in the top right corner of the video display: video frame rate, CPU utilization and memory usage. The live video display is a floating window which can be repositioned and is resizable.

The main program loop converts the sound source position from world to head coordinates, extracts the appropriate HRIR vectors and gain values and writes the coefficient values to the correct memory addresses on the DSP board via serial port.

6. RESULTS

The system was demonstrated using an overhead projector to display the Windows control panel and the live video display of Figure 6. A PS3Eye camera was mounted on a photo tripod underneath and in front of the projector display and an area of around 15 or 20 feet was cleared in front of the display so that the participants could move freely around to experience the sound localization effects. A stereo extension cable was used with the headset to enable the participants to roam about the area in front of the display. In general, all participants reported good sound localization effects, although sometimes after a little experimentation by selecting different HRTF subject files. We were able to place a virtual sound source at some specific position in the room and the participants were able to detect the sound “hanging” at some position in space, moving around and then placing their hand at the location the sound appeared to be coming from.

7. CONCLUSION

The real-time DSP approach to simulating human sound localization is both effective and inexpensive. The DSP processor can be dedicated to performing the computationally intensive task of implementing the two 200-tap FIR filters, freeing the CPU to perform the (also computationally intensive) head tracking and other functions.

Figure 6 The Windows program GUI and video display.

The addition of head tracking capability dramatically enhances the believability of the virtual 3D sound source. This system is a flexible demonstration platform which can be used for further development of specific 3D sound applications.

8. REFERENCES
