A Real-Time 3D Audio Simulator for Cognitive Hearing Science

Mark Wickert[‡][*]

http://www.youtube.com/watch?v=dhRUe-gz690

Abstract—This paper describes the development of a 3D audio simulator for use in cognitive hearing science studies and also for general 3D audio experimentation. The framework that the simulator is built upon is `pyaudio_helper`, which is a module of the package `scikit-dsp-comm`. The simulator runs in a Jupyter notebook and makes use of Jupyter widgets for interactive control of audio source positioning in 3D space. 3D audio has application in virtual reality and in hearing assistive devices (HAD) research and development. At its core the simulator uses digital filters to represent the sound pressure wave propagation path from the sound source to each ear canal of a human subject. Digital filters of 200 coefficients each for left and right ears are stored in a look-up table as a function of azimuth and elevation angles of the impinging sound's source.

Index Terms—Head-related impulse response (HRIR), Head-related transfer function (HRTF), binaural hearing, virtual reality, audiology, hearing assistive devices (HAD),

Introduction

In cognitive hearing science binaural hearing models how sound pressure waves arrive at either ear drum, at the end of the ear canal, or in the case of typical measurements, at the entry to the ear canal, both as a function of the arrival angle in 3D (azimuth and elevation) and radial distance. A tutorial on 3-D audio can be found at [HCI]. This leads to the need for the head related impulse response (HRIR) (time-domain) or head-related transfer function (HRTF) (frequency domain) for a particular human subject. Traditionally human subjects are placed in an anechoic chamber with a sound source placed at e.g. one meter from the head and then moved relative the subject's head over a range of azimuth and elevation angles, with the HRIR measured at each angle. The 3D simulator described here uses a database of HRIR's from the University of California, Davis, originally in the Center for Image Processing and Integrated Computing (CIPIC), [CIPICHRTF], to describe a given subject. In the `pyaudio_helper` application the HRIR at a given angle is represented by two (left and right ear) 200 coefficient digital filters that the sound source audio is passed through. Here the data base for each subject holds 25 azimuth and 50 elevation angles to approximate continuous sound source 3D locations.

* Corresponding author: mwickert@uccs.edu
‡ University of Colorado Colorado Springs

Obtaining individual HRTFs is a challenge in itself and the subject of much research.

In a related research project *deep learning* is being investigated as a means to fit a human subject to the CIPIC HRTF database of subjects, based on 27 upper torso anthropometrics (measurements) of the subject. As a simple solution, we can also consider using a simple spherical head model, and its corresponding HRTF, which makes use of spherical harmonics to solve for the sound pressure magnitude and phase at any location on the sphere surface. A frequency sweep of magnitude and phase is then inverse Fourier transformed to obtain the HRIR. The ultimate intent of the simulator is to serve as a clinical tool for blind sound source localization experiments. Human subjects will be exposed to several different HRIR models, where at least one model is a *personalized fit* based on deep learning using anthropometrics and/or a finite element wave equation solution using a 3D rendering of the subject's shoulders and head. 3D rendering of a subject can be obtained using *photogrammetry*, which estimates three-dimensional coordinates of points on an object from a collection of photographic images taken from different positions.

3D Geometry

To produce a synthesized 3D audio sound field, we start with a geometry where the center of the coordinate frame is the intersection between the subject's *mid-sagittal* or vertical *median plane* and the line connecting the left and right ear canals. This is referred to as being *head-centered*. The coordinate systems used in this paper are shown in Figure 1. The primary head-centered system has cartesian coordinates labeled (x,y,z) and associated cylindrical coordinates $(r_{xy},\phi_{\text{az}},h_y)$ (black labels in Figure 1). The cylindrical coordinates will be used in Jupyter notebook apps presented later as the interface for GUI controls to conveniently position the audio source about a subject's head. A secondary head-centered system, used by CIPIC, has cartesian coordinates labeled (x_1,x_2,x_3) and associated spherical coordinates (r,ϕ,θ) (purple labels in Figure 1). The first coordinate system is motivated by [Fitzpatrick], and its usage is explained in detail in the section FIR Filter Coefficient Set Selection. The second system is referred to by CIPIC as the *interaural-polar coordinate system* (IPCS), which is used to index into the HRIR filter pairs which produce the right and left audio outputs.

The 3D audio rendering provided by the simulator developed in this paper relies on the 1250 HRIR measurements taken using the geometrical configuration shown in Figure 2. A total of 45 subjects are contained in the CIPIC HRIR database, both human

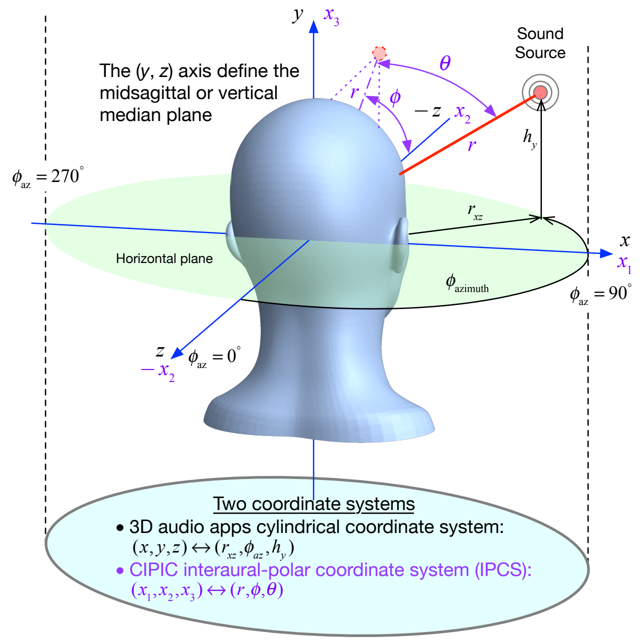

The (y, z) axis define the midsagittal or vertical median plane
Sound Source
Horizontal plane
Two coordinate systems
• 3D audio apps cylindrical coordinate system: $(x,y,z) \leftrightarrow (r_{xz},\phi_{az},h_y)$
• CIPIC interaural-polar coordinate system (IPCS): $(x_1,x_2,x_3) \leftrightarrow (r,\phi,\theta)$
$\phi_{az}=270^\circ$
$\phi_{az}=90^\circ$
$\phi_{az}=0^\circ$

Fig. 1: The primary head-centered coordinate system, (x,y,z), used in the 3D audio simulator, along with the secondary system, (x_1,x_2,x_3) used by CIPIC via IPCS and spherical coordinates (r,ϕ,θ).

and the mannequin KEMAR (Knowles Electronics Manikin for Auditory Research) [CIPICHRTF]. For subject 165 in particular, the left-right channel HRIR is shown in Figure 3, for a particular cylindrical coordinate system triple (r_{xz},h_y,ϕ_{az}). Figure 3 in particular illustrates two binaural cues, *interaural level difference* ILD and *interaural time difference* ITD, that are used for accurate localization of a sound source. With $\phi_{az}=130^\circ$ we see as expected, the impulse response for the right ear arriving ahead of the left ear response, and with greater amplitude.

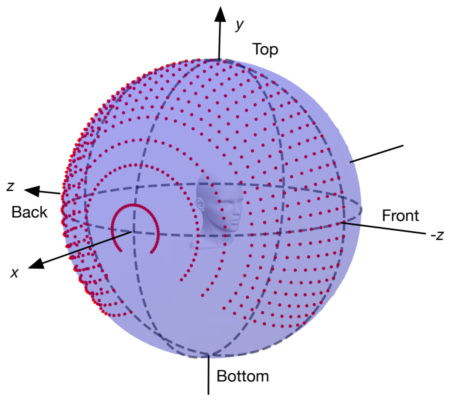


Fig. 2: The CIPIC audio source locations, effectively on a 1 m radius sphere, used to obtain 1250 HRIR measurements for each of 45 subjects (only the right hemisphere locations shown).

Real-Time Signal Processing

In this section we briefly describe the role real-time digital signal processing (DSP) plays in implementing the 3D audio simulator. A top level block diagram of the 3D audio simulator is shown in Figure 4. For an audio source positioned at (x,y,z) relative to the head center, the appropriate HRIR right and left channel digital

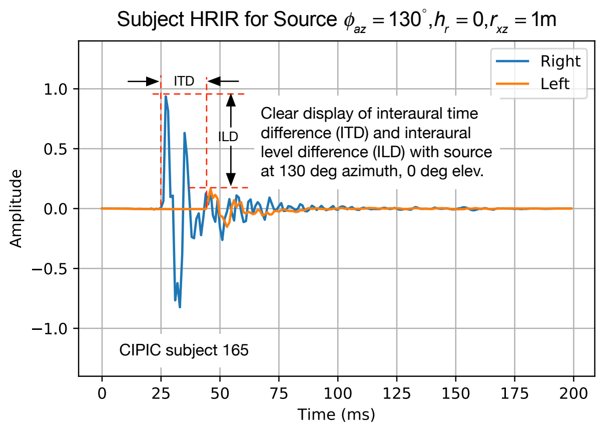


Fig. 3: Example right/left HRIR plots for a particular arrival angle pulled from CIPIC for subject 165.

filter coefficients are utilized along with gain scaling to account for radial distance relative to 1 m and a parallax correction factor. Gain scaling and parallax correction, are taken from [Fitzpatrick], and are explained in more detail in the following section of this paper.

To implement the filtering action we use the `pyaudio_helper` framework [Wickert] of Figure 5, which interfaces to the audio subsystem of a personal computer. The framework supports real-time signal processing, in particular filtering using core signal processing functions of `scipy.signal` [ScipySignal]. The 200 coefficients of the right and left HRIR are equivalent to the coefficients in a finite impulse response (FIR) digital filter which produce a discrete-time output signal or sequence $y_R[n]/y_L[n]$ from a single audio source signal $x[n]$. All of the signals are processed with at a sampling rate of $f_s=44.1$ kHz, as this is rate used in forming the CIPIC database. In mathematical terms we have the output signals that drive

$$y_R[n] = G_R \sum_{m=0}^{M} b_R x[n-m] \quad (1)$$

$$y_L[n] = G_L \sum_{m=0}^{M} b_L x[n-m] \quad (2)$$

where G_R and G_L are right/left gain scaling factors that take into account the source distance relative to the 1 m distance used in the CIPIC database and b_R and b_L are the right/left HRIR coefficient sets appropriate for the source location.

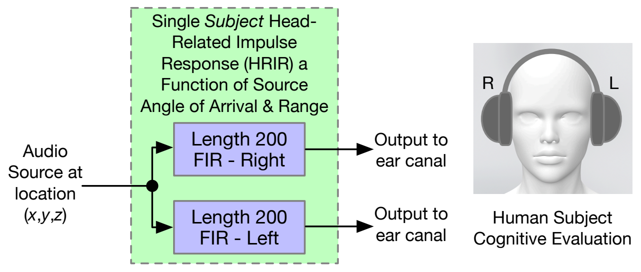

Single *Subject* Head-Related Impulse Response (HRIR) a Function of Source Angle of Arrival & Range
Length 200 FIR - Right
Length 200 FIR - Left
Audio Source at location (x,y,z)
Output to ear canal
Output to ear canal
R
L
Human Subject Cognitive Evaluation

Fig. 4: Real-time DSP filtering with coefficients determined by the audio source (x,y,z) location.

To produce real-time filtering with `pyaudio_helper` requires [Wickert] (i) create an instance of the `DSP_io_stream`

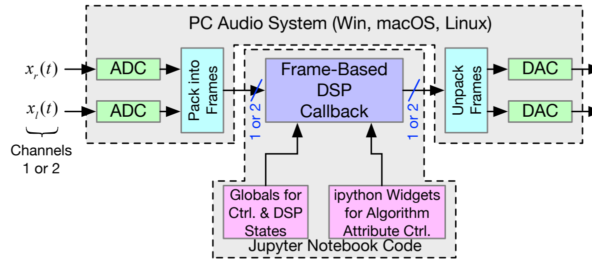


Fig. 5: *The pyaudio_helper framework for real-time DSP in the Jupyter notebook.*

class by assigning valid PC audio input and output device ports to it, (ii) define a `callback` function to process the input signal sample frames into right/left output sample frames according to (1), and (iii) call the method `interactive_stream()` to start streaming. All of the code for the 3D simulator is developed in a Jupyter notebook for prototyping ease. Since [Wickert] details steps (i)-(iii), in the code snippet below we focus on the key filtering expressions in the callback and describe the playback of a geometrically positioned *noise* source via headphones:

```
def callback(in_data, frame_length, time_info,
             status):
    global ...
    ...
    #************************************************
    # DSP operations here:
    # Apply Kemar HRIR left and right channel
    # filters at the sound source location in
    # cylindrical coordinates mapped to cartesian
    # coordinates from GUI sliders
    # The input to both filters comes by first
    # combining x_left & x_right channels or here
    # input white noise
    x_mono = Gain.value*5000*randn(frame_length)
    subj.cart2ipcs(r_xz_plane.value*sin(pi/180* \
                   azimuth.value), #x
                   y_axis.value,   #y
                   r_xz_plane.value* \
                   cos(pi/180* \
                   azimuth.value)) #z
    # Filter a frame of samples and save initial
    # conditions for the next frame
    y_left, zi_left = signal.lfilter(subj.coeffL,
                                     1,subj.tL*x_mono,
                                     zi=zi_left)
    y_right, zi_right = signal.lfilter(subj.coeffR,
                                     1,subj.tR*x_mono,
                                     zi=zi_right)
    #************************************************
    ...
    # Convert ndarray back to bytes
    return y.tobytes(), pah.pyaudio.paContinue

# Create a ss_mapping2CIPIChrir object
# SUBJECT 20, 21 (KEMAR SM ears),
# & 165 (KEMAR LG ears)
# subject_200, 201 is 8.75 cm, 10 cm sphere
subj = ss_mapping2CIPIChrir('subject_165')
# Initialize L/R filter initial conditions
zi_left = signal.lfiltic(subj.coeffL,1,[0])
zi_right = signal.lfiltic(subj.coeffR,1,[0])
# Create a IO stream object and start streaming
DSP_IO = pah.DSP_io_stream(callback,0,1,
                           frame_length=1024,
                           fs=44100,Tcapture=0)
DSP_IO.interactive_stream(0,2)
# Show Jupyter widgets
widgets.HBox([Gain,r_xz_plane,azimuth,y_axis])
```

FIR Filter Coefficient Set Selection

To finally render 3D audio requires selection of the appropriate right/left filter coefficient set, and if needed range correction. For the special case of the source on the 1 m CIPIC reference sphere, we simply pick the coefficient set that lies closest to the desired IPCS angle pair (ϕ, θ).

For the more typical case of the source range, $r = \sqrt{x^2+y^2+z^2} \neq 1$, more processing is required. The approach taken here follows the methodology of [Fitzpatrick] by using the primary cartesian coordinates of Figure 1 to additionally perform *parallax* correction and source range amplitude correction. At distance r from a point source the sound wave energy diverges by $1/r^2$, so in terms of wave amplitude we include a scale factor of $1/r$. Here the inverse distance correction also takes into account the fact that the entry to the ear canal is offset from the head center by the mean head radius R. The second correction factor is *parallax*, which is graphically depicted in Figure 6 for the special case of a source in the horizontal plane and directly in front of the head. Both corrections are addressed in detail in [Fitzpatrick]. For a source not on the unit sphere, sound parallax requires an adjustment in the HRIR coefficients, unique to the right and left ears. If we extend rays from the right and left ears that pass through the sound source location and then touch the unit sphere, the required azimuth values will be shifted to locations on either side of the true source azimuth. The corresponding HRIR values where these rays contact the unit sphere, respectively, perform the needed parallax correction. The actual database entries utilized are those that are closest to the intersection points.

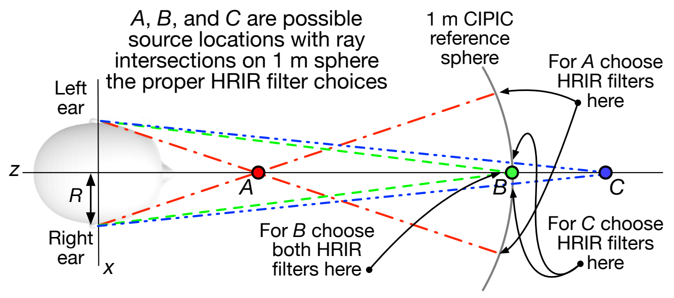

A, B, and C are possible source locations with ray intersections on 1 m sphere the proper HRIR filter choices
1 m CIPIC reference sphere
For A choose HRIR filters here
Left ear
z
R
A
B
C
Right ear
x
For B choose both HRIR filters here
For C choose HRIR filters here

Fig. 6: *Parallax correction geometry for three possible source locations in the horizontal plane: $A < 1$ m, $B = 1$ m, and $C > 1$ m, directly in front of the head.*

The class `ss_mapping2CIPIChrif()` takes the source location, (x,y,z), and using the single method `cart2ipcs(self,x,y,z)`, produces the parallax corrected right and left HRIR filter coefficients and range amplitude scaling factors. The code is listed below:

```
class ss_mapping2CIPIChrir(object):
    """
    A class for sound source mapping to the CIPIC
    HRIR database

    CIPIC uses the interaural polar coordinate
    system (IPCS). The reference sphere for the
    head-related transfer function (HRTF)
    measurements/head-related impulse response
    (HRIR) measurements has a 1m radius.

    Mark Wickert June 2018

def __init__(self,sub_foldername,
             head_radius_cm = 8.75):
    """
    Object instantiation

    The default head radius is 8.75 cm
    """
    # Store the head radius in meters
    self.head_radius = head_radius_cm/100

    # Store the HRIR 200 tap FIR filter coef sets
    self.subject = sub_foldername
    hrir_LR = io.loadmat( self.subject + \
                      '/hrir_final.mat')
    self.hrirL = hrir_LR['hrir_l']
    self.hrirR = hrir_LR['hrir_r']

    # Create LUTs for the azimuth and elevation
    # values. This will make it easy to quantize
    # a given source location to one of the
    # available HRIRs in the database.
    self.Az_LUT = np.hstack(([-80,-65,-55],
                   np.arange(-45,45+5,5.0),
                              [55,65,80]))
    self.El_LUT = -45 + 5.625*np.arange(0,50)

    # Initialize parameters
    self.tR = 1 # place source on unit sphere
    self.tL = 1 # directly in front of listener
    self.elRL = 0
    self.azR = 0
    self.azL = 0
    self.AzR_idx = 0
    self.AzL_idx = 0
    self.ElRL_idx = 0

    # Store corresponding right and left ear FIR
    # filter coefficients
    self.coeffR = self.hrirR[0,0,:]
    self.coeffL = self.hrirL[0,0,:]


def cart2ipcs(self,x,y,z):
    """
    Map cartesian source coordinates (x,y,z) to
    the CIPIC interaural polar coordinate system
    (IPCS) for easy access to CIPIC HRIR. Parallax
    error is also dealt with so two azimuth values
    are found. To fit IPCS the cartesian
    coordinates are defined as follows:

    (0,0,0) <--> center of head.
    (1,0,0) <--> unit vector pointing outward from
                 the right on a line passing from
                 left to right through the left
                 and right ear (pinna) ear canals
    (0,1,0) <--> unit vector pointing out through
                 the top of the head.
    (0,0,1) <--> unit vector straight out through
                 the back of the head, such that
                 a right-handed coordinate system is
                 formed.

    Mark Wickert June 2018, updated June 2019
    """
    # First solve for the parameter t, which is used
    # to describe parametrically the location of the
    # source at (x,y,z) on a line connecting the
    # right or left ear canal entry point to the
    # unit sphere.

    # The right ear (pinna) solution
    aR = (x-self.head_radius)** + y**2 + z**2
    bR = 2*self.head_radius*(x-self.head_radius)
    cRL = self.head_radius**2 - 1
    # The left ear (pinna) solution
    aL = (x+self.head_radius)**2 + y**2 + z**2
    bL = -2*self.head_radius*(x+self.head_radius)

    # Find the t values which are also the gain
    # values to be applied to the filter.
    self.tR = max((-bR+np.sqrt(bR**2-4*aR*cRL)) \
               /(2*aR),
              (-bR-np.sqrt(bR**2-4*aR*cRL))/(2*aR))
    self.tL = max((-bL+np.sqrt(bL**2-4*aL*cRL)) \
               /(2*aL),
              (-bL-np.sqrt(bL**2-4*aL*cRL))/(2*aL))
    # Find the IPCS elevation angle and mod it
    elRL = 180/np.pi*np.arctan2(y1,-z1)
    if elRL < -90:
          elRL += 360
    self.elRL = elRL
    self.azR = 180/np.pi* \
                  np.arcsin(np.clip(self.head_radius\
                   + self.tR*(x1-self.head_radius),
                   -1,1))
    self.azL = 180/np.pi* \
                  np.arcsin(clip(-self.head_radius\
                   + self.tL*(x1+self.head_radius),
                   -1,1))
    # Find closest database entry in Az & El
    self.AzR_idx = np.argmin((self.Az_LUT \
                          - self.azR)**2)
    self.AzL_idx = np.argmin((self.Az_LUT \
                          - self.azL)**2)
    self.ElRL_idx = np.argmin((self.El_LUT \
                          - self.elRL)**2)
    self.coeffR = self.hrirR[self.AzR_idx,
                          self.ElRL_idx,:]
    self.coeffL = self.hrirL[self.AzL_idx,
                          self.ElRL_idx,:]
```

In the `__init__` method all the right left filter coefficients for the chosen subject database entry are copied into class attributes and look-up tables (LUTs) are populated in terms of IPCS angles to ease selecting the needed right/left filters. Note in particular the scale factors `self.tR` and `self.tL` are the inverse distance wave amplitude correction factors representing G_R and G_L in (1) and (2), respectively.

3D Audio Simulator Notebook Apps

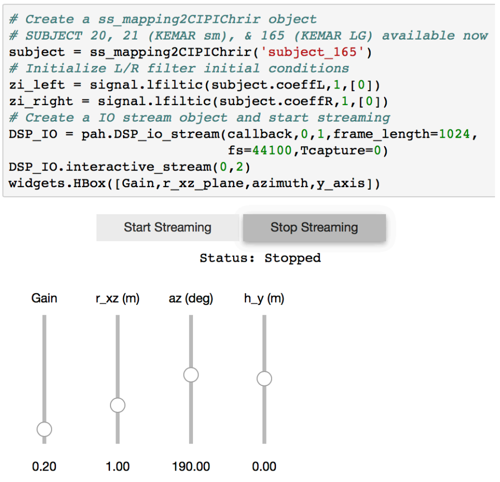
For human subject testing and general audio virtual reality experiments, two applications (apps) that run in the Jupyter notebook were created. The first allows the user to *statically* locate an audio source in space, while the second creates a *time-varying motion* audio source. For human subject tests the static source is of primary interest. Both apps have a GUI slider interface that use the cylindrical coordinates described in Figure 1 to control the position the source.

Static Sound Source

The first and foremost purpose of the 3D audio simulator is to be able to statically position an audio source and then ask a human subject where the source is located (localization). This is a cognitive experiment, and can serve many purposes. One purpose in the present research is to to see how well the HRIR utilized in the simulator matches the subject's true HRIR. As mentioned in the introduction, an ongoing study is to estimate an *individualized HRIR* using deep machine learning/deep learning. The Jupyter Widgets slider interface for this app is shown in Figure 7

Dynamic Sound Source Along a Trajectory

From a virtual reality perspective, we were also interested in giving a subject a moving sound source experience via headphones.



```
# Create a ss_mapping2CIPIChrir object
# SUBJECT 20, 21 (KEMAR sm), & 165 (KEMAR LG) available now
subject = ss_mapping2CIPIChrir('subject_165')
# Initialize L/R filter initial conditions
zi_left = signal.lfiltic(subject.coeffL,1,[0])
zi_right = signal.lfiltic(subject.coeffR,1,[0])
# Create a IO stream object and start streaming
DSP_IO = pah.DSP_io_stream(callback,0,1,frame_length=1024,
                           fs=44100,Tcapture=0)
DSP_IO.interactive_stream(0,2)
widgets.HBox([Gain,r_xz_plane,azimuth,y_axis])
```

Fig. 7: *Jupyter notebook for static positioning of the audio test source.*

In this case we consider an *orbit like* sound source trajectory. The trajectory as shown in Figure 8, is a circular orbit with parameters of roll, pitch, and hight, relative to the ear canal centerline. The Jupyter Widgets slider interface for this app is shown in Figure 9.

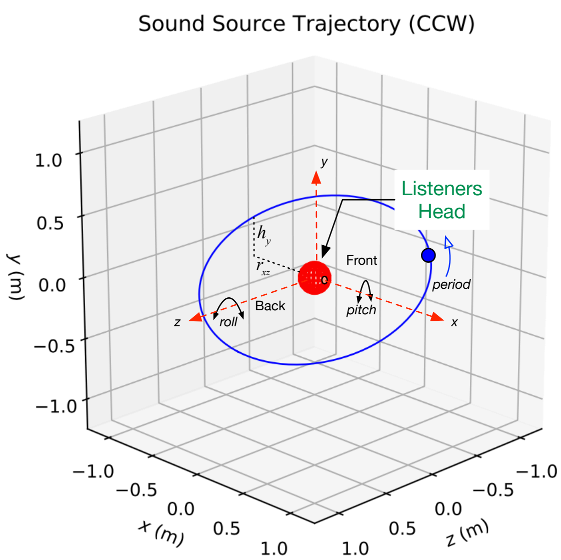


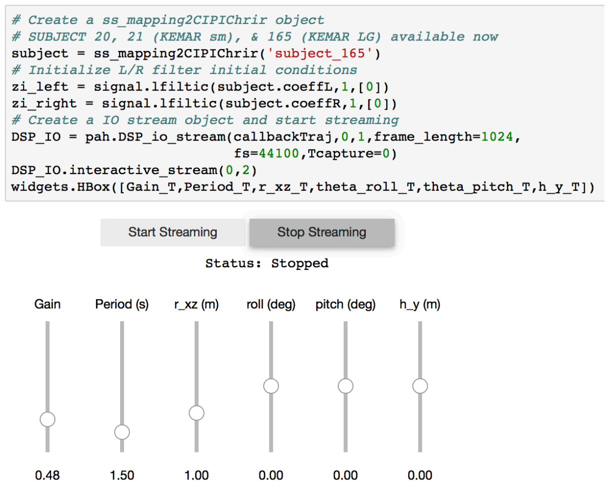
Fig. 8: *The sound source trajectory utilized in the dynamic sound source app.*

Spherical Head Model as a Simple Reference HRIR

In blind testing of human subjects it is also of interest to offer other HRIR solutions, e.g., the [KEMAR] mannequin head or a simple spherical head [Duda] and [Bogelein]. In this section we consider a spherical head model with the intent of using the results of [Duda] to allow the construction of a CIPIC-like database entry, that can be used in the 3D audio simulator described earlier in this paper.

General Pressure Wave Solution

As a starting point, the acoustics text [Beranek], provides a solution for the resultant sound pressure at any point in space when a sinusoidal plane wave sound pressure source impinges



```
# Create a ss_mapping2CIPIChrir object
# SUBJECT 20, 21 (KEMAR sm), & 165 (KEMAR LG) available now
subject = ss_mapping2CIPIChrir('subject_165')
# Initialize L/R filter initial conditions
zi_left = signal.lfiltic(subject.coeffL,1,[0])
zi_right = signal.lfiltic(subject.coeffR,1,[0])
# Create a IO stream object and start streaming
DSP_IO = pah.DSP_io_stream(callbackTraj,0,1,frame_length=1024,
                           fs=44100,Tcapture=0)
DSP_IO.interactive_stream(0,2)
widgets.HBox([Gain_T,Period_T,r_xz_T,theta_roll_T,theta_pitch_T,h_y_T])
```

Fig. 9: *Jupyter notebook for setting the parameters of a sound source moving along a trajectory with prescribed motion characteristics.*

upon a rigid sphere of radius R, centered at the coordinate system origin. Rotationally symmetric spherical coordinates, r and θ are appropriate here. First consider the incident plane wave $\tilde{p}_I(r,\theta)$, in the expansion

$$\tilde{p}_I(r,\theta_i) = \tilde{p}_0 \sum_{n=0}^{\infty} (-j)^n (2n+1) j_n(kr) P_n(\cos\theta_i), \tag{3}$$

where θ_i is the incidence angle between the plane wave and measurement point, $P_m(x)$ is the nth-order Legendre polynomial, $j_n(x)$ is the nth-order spherical Bessel function of the first kind, $k = 2\pi f/c$ is the wavenumber, with f frequency in Hz and $c = 344.4$ m/s the propagation velocity in air. We set the incident wave complex pressure $\tilde{p}_0 = 1\angle 0°$ for convenience.

Finally, solve for the scattered wave, $\tilde{p}_s(r,\theta_i)$, by applying boundary conditions, see [Beranek] for details. The resultant wave is the sum of the incident and scattered waves as given below:

$$\begin{aligned}\tilde{p}(r,\theta_i) &= \tilde{p}_I(r,\theta_i) + \tilde{p}_s(r,\theta_i) \\ &= \sum_{n=0}^{\infty} (-j)^n (2n+1) P_n(\cos\theta_i) \\ &\quad \cdot \left[j_n(kr) - \frac{j_n'(kR)}{h_n'^{(2)}(kR)} h_n^{(2)}(kr) \right]\end{aligned} \tag{4}$$

where $j_n'(x)$ is the spherical Bessel function of the first kind derivative, $h_n^{(2)}(kr)$ is the nth-order spherical Hankel function of the second kind, and $h_n'^{(2)}(kr)$ is the corresponding derivative. Figure 10 shows the pressure magnitude at 2000 Hz for $R = 8.75$ cm, for the plane wave traveling along the $+z-$axis. For plotting convenience, the axes z and $w = \sqrt{x^2+y^2}$ are cylindrical coordinates, as the sphere has axial symmetry. To be clear z and w are related to the original spherical coordinates of the math model by $r = \sqrt{w^2+z^2}$ and $\cos\theta_i = z/\sqrt{w^2+z^2}$.

The calculations required to evaluate (4), and thus create the plot of Figure 10, conveniently make use of functions in `scipy.special`. This is shown in the code listing below:

```
def pS(w, z, f, R = 0.0875, N = 50):
    """
    Scattered field from a rigid sphere

    w = radial comp in cylind coord
```

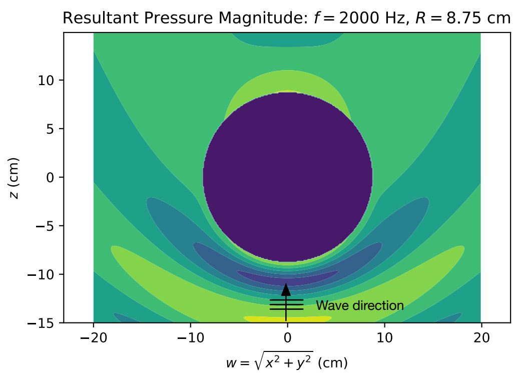


Fig. 10: *The resultant sound pressure wave magnitude in cylindrical coordinates z and w, due to scattering of a plane wave from a rigid sphere.*

```
    z = axial comp in cylind coord
    f = frequency in Hz
    R = sphere radius in m
    N = summation upper boundary

    p_polar = pressure in Pa for p0 = 1 Pa

    Mark Wickert November 2018
    """
    p_0 = 1
    k = 2*pi/(344.4/f)
    p_polar = zeros((len(z),len(w)),
                    dtype=complex128)
    for n,wn in enumerate(w):
        for m,zm in enumerate(z):
            r = sqrt(zm**2 + wn**2)
            cos_theta = zm/sqrt(zm**2 + wn**2)
            for kk in range(N+1):
                if r <= R:
                    p_polar[m,n] = 0.0
                else:
                    p_polar[m,n] += p_0*(-1j)**kk *\
                    (2*kk+1) * \
                    special.spherical_jn(kk,
                    k*R,True)/spherical_hn2(kk,
                    k*R,True) * spherical_hn2(kk,
                    k*r) * \
                    special.lpmv(0,kk,cos_theta)
    return -p_polar

def spherical_hn2(n,z,derivative=False):
    """ Spherical Hankel Function 2nd Kind """
    return special.spherical_jn(n,z,deriv=False) \
           -1j * special.spherical_yn(n,z,
           derivative=False)
```

The use of $R = 8.75$ cm is motivated by the *standard head* radius discussed in [Duda]. It is interesting to note that there is a *bright spot* on the back side ($\theta_i = 180°$) due to constructive interference between the waves traveling around either side of the sphere.

HRTF on the Sphere Surface

In signal processing, the *transfer function*, $H(f) = |H(f)|e^{j\angle H(f)}$, is a ratio of two complex numbers as a function frequency in Hz. In the denominator we have the magnitude and phase (angle) of the sinusoidal signal input to a system and in the numerator we have the magnitude and phase of the corresponding output signal (measurement point on the sphere or ultimately the ear canal). For the case of the HRTF the output is the sound pressure magnitude and phase at the entrance to the right and left ear canals. In the case of the CIPIC database the location of the source is at a particular azimuth and elevation on a 1 m sphere centered over the head. The HRTF of a sphere is defined more generally as the output can be any point on the surface of the sphere. The input location is generally at some distance r from the center of the sphere.

In [Duda] the HRTF is defined as the ratio of the sound pressure on the surface of the sphere divided by the pressure at the sphere center, given that the sphere *is not* present:

$$H(\theta_i, f, r, R) = \frac{r}{kR^2}e^{jkr}\sum_{n=0}^{\infty}(2n+1)P_n(\cos\theta_i)\frac{h_n^{(2)}(kr)}{h_n'^{(2)}(kR)},\ r > R \quad (5)$$

where θ_i is the angle of incidence between the source and measurement point, f is the operating frequency in Hz, r is the distance from the source to the center of the sphere, and once again R is the sphere radius. Recall also that the wave number k contains f.

Formally this transfer function definition should include the propagation delay time from the source location r to the sphere center, but this is a *linear phase* of the form $\exp(-j2\pi fr/c)$ that can be dealt with as a time shift once the inverse Fourier transform is used to obtain the HRIR. Later we set $r = 1$ m to match the CIPIC source location relative to the head center.

An efficient algorithm for the calculation of (5) is presented in [Duda], requiring no special functions as a result of using special function recurrence relationships. The Python implementation, shown below, also incorporates an error threshold for terminating the series approximation:

```
def HRTF_sph(theta, f, r = 1.0, R = 0.01, c = 344.4,
             threshold = 1e-6):
    """
    HRTF calculation for a rigid sphere with source
    r meters from the sphere center

    Coded from pseudo-code to Python by Mark Wickert

    Reference: Appendix A of J. Acoust. Soc. Am.,
    Vol. 104, No. 5, November 1998 R. O. Duda and
    W. L. Martens: Range dependence of the response
    of a spherical head model.
    """
    x = np.cos(theta*np.pi/180)
    mu = (2 * np.pi * f * R)/c
    rho = r/R
    zr = 1/(1j * mu * rho)
    zR = 1/(1j * mu)
    Qr2 = zr
    Qr1 = zr * (1 - zr)
    QR2 = zR
    QR1 = zR * (1 - zR)
    P2 = 1
    P1 = x
    summ = 0
    term = zr/(zR * (zR - 1))
    summ += term
    term = (3 * x * zr * (zr - 1) )/ \
           (zR * (2 * zR * (zR - 1) + 1))
    summ += term;
    oldratio = 1
    newratio = np.abs(term)/np.abs(summ)
    m = 2
    while (oldratio > threshold) or \
          (newratio > threshold):
        Qr = -(2 * m - 1) * zr * Qr1 + Qr2
        QR = -(2 * m - 1) * zR * QR1 + QR2
        P = ((2 * m - 1) * x * \
             P1 - (m - 1) * P2)/m
        term = ((2 * m + 1) * P * Qr)/((m + 1) \
               * zR * QR - QR1)
        summ += term
        m += 1
        Qr2 = Qr1
        Qr1 = Qr
        QR2 = QR1
        QR1 = QR
        P2 = P1
        P1 = P
        oldratio = newratio
        newratio = np.abs(term)/np.abs(summ)
    # conjugate to match traveling wave convention
    H = np.conj((rho * np.exp(-1j * mu) * summ)/\
                (1j * mu))
    return H
```

HRIR on the Sphere Surface

The next step is to calculate the impulse response $h(t)$ corresponding to $H(f)$ via the inverse Fourier transform of the HRTF. Since we are working with digital (discrete-time) signal processing, the inverse discrete Fourier transform (IDFT) is used here, as opposed to the Fourier integral. We take samples of the HRTF at uniformly spaced frequency samples, Δf, running from 0 to one half the CIPIC sampling rate, $f_s = 44.1$kHz. This makes $h(t) \to h(n/f_s) = h[n]$ in the Python implementation shown below:

```
def freqr2imp(H,win_att = 100):
    """
    Transform the frequency response of a real
    impulse response system back to the impulse
    response, with smoothing using a window
    function.

    Mark Wickert, May 2019
    """
    Nmax = len(H)
    if win_att == 0:
        h = np.fft.irfft(H)
    else:
        W = signal.windows.chebwin(2*Nmax,
                     win_att,sym=True)[Nmax:]
        h = np.fft.irfft(H*W)
    return h


def compute_HRIR(theta_deg, r = 1.0, R = 0.0875,
          fs = 44100, roll_factor = 20):
    """
    HRIR for rigid sphere at incidence angle
    theta_deg, distance r and radius R using
    sampingrate fs Hz

    Mark Wickert, June 2019
    """
    fs = 44100
    Nfft = 2**10
    df = fs/Nfft
    f = np.arange(df,fs/2,df)
    df = fs/Nfft
    f = np.arange(df,fs/2,df)
    HRTF = np.zeros(len(f),dtype=np.complex128)
    for k, fk in enumerate(f):
        HRTF[k] = HRTF_sph(theta_deg,fk,r=r,R = R)
    # Set DC value to 1
    HRTF = np.hstack(([1],HRTF))
    f = np.hstack(([0],f))

    HRIR = freqr2imp(HRTF,win_att=100)
    # Scale HRIR so the area is unity
    G0 = 1/(np.sum(HRIR)*1/fs)
    t = np.arange(len(HRIR))/fs*1000
    return t, np.roll(G0*HRIR,roll_factor)
```

We choose Δf to obtain at least 100 samples on $[0, f_s/2]$, so that when `np.fft.irfft()` is employed, the full real impulse response length will be 200. The function $freq2imp()$ also includes frequency domain windowing, via `signal.windows.chebwin()` to provide some smoothing to the discrete-time approximation. In Figure 11 we show a collection of HRIR plots, created using `HRTF_sph()`, for the source 1 m away from the center of a 8.75 cm radius sphere.

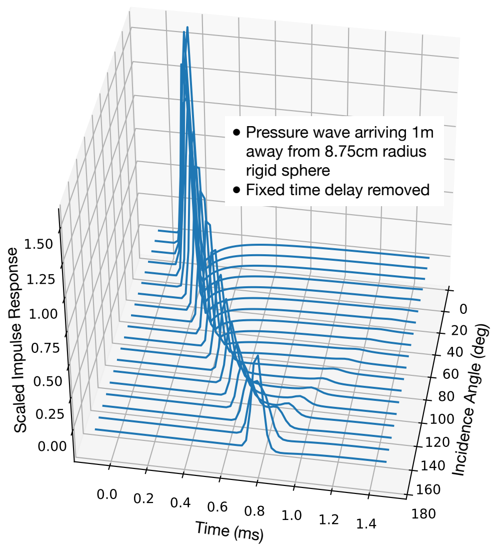


Fig. 11: *Using the spherical harmonics formulation of [Duda] to obtain the HRTF and then the HRIR as a function of sound source incidence angle from* 0° *to* 180°.

Building a CIPIC Database Entry

To finally create a CIPIC-like database entry for a spherical head, we have to relate the angle of incidence in the HRTF expression (5) to the angle of arrival of an audio source on the CIPIC 1 m sphere of Figure 2, relative to right and left ear canal entries at $\phi_{az} = \pm 80°$ (a set back of $\pm 100°$ from the front). The problem is depicted in Figure 12. This problem turns out to be a familiar analytic geometry problem, that of finding the angle between two 3D vectors passing through the origin, e.g.

$$\theta_{\vec{S}\vec{R}} \quad = \quad \cos^{-1}\left(\frac{\vec{S}\cdot\vec{R}}{|\vec{S}|\,|\vec{R}|}\right) = x_S \sin\phi_R + z_S \cos\phi_R \qquad (6)$$

where $\vec{R}$ is the vector to the right ear canal with angle ϕ_R, assumed to lie in the horizontal plane, and $\vec{S}$ is the vector to the source of length 1 m with primary coordinate system components (x_S, y_S, z_S) as defined in Figure 1. A similar relation holds for the left ear canal entry.

We need to fill the database using the CIPIC angle of arrival source grid using the secondary (ICPS) coordinate system. The coordinate conversion between x_S and z_s and the IPCS is $x_s = r\sin\theta_{\text{CIPIC}}$ and $z_s = -r\cos\phi_{\text{CIPIC}}\cos\theta_{\text{CIPIC}}$, so with $r = 1$ the angle of incidence formula (6) in final form is

$$\theta_{\vec{S}\vec{R}} = \sin\theta_{\text{CIPIC}}\sin\phi_R - \cos\phi_{\text{CIPIC}}\cos\theta_{\text{CIPIC}}\cos\phi_R \qquad (7)$$

and similarly for the left ear canal.

The steps for producing the HRIR filter pair over 1250 IPCS angle pairs is summarized in Figure 13.

Finally putting this all together, code was written in a Jupyter notebook to generate a CIPIC-like database entry, using `scipy.io` to write a MATLAB `mat` file, e.g., `subject_200` is a spherical head, with no ears (pinna), containing two HRIR arrays:

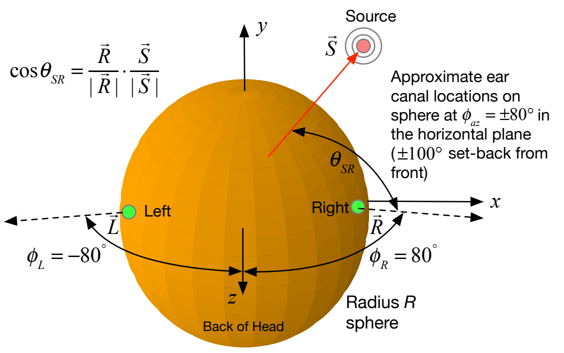

$\cos\theta_{SR} = \frac{\vec{R}}{|\vec{R}|} \cdot \frac{\vec{S}}{|\vec{S}|}$
y
Source
$\vec{S}$
Approximate ear canal locations on sphere at $\phi_{az} = \pm 80°$ in the horizontal plane ($\pm 100°$ set-back from front)
θ_{SR}
Left
Right
x
$\vec{L}$
$\vec{R}$
$\phi_L = -80°$
$\phi_R = 80°$
z
Radius R sphere
Back of Head

Fig. 12: Solving for the angle between the source and a ray extending from the right and left ears, also showing a set back of the ear canal by $\pm 100°$ *from the from the font of the head.*

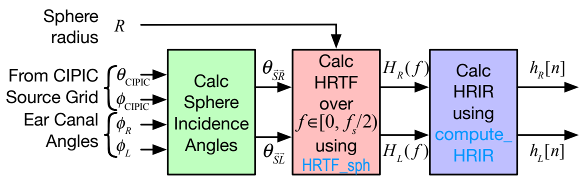

Sphere radius
R
From CIPIC Source Grid
θ_{CIPIC}
ϕ_{CIPIC}
Ear Canal Angles
ϕ_R
ϕ_L
Calc Sphere Incidence Angles
$\theta_{\vec{S}\vec{R}}$
$\theta_{\vec{S}\vec{L}}$
Calc HRTF over $f \in [0, f_s/2)$ using HRTF_sph
$H_R(f)$
$H_L(f)$
Calc HRIR using compute_HRIR
$h_R[n]$
$h_L[n]$

Fig. 13: A block diagram depicting the steps involved in calculating the HRIR right and left channel impulse responses, $h_R[n]$ *and* $h_L[n]$*, starting from CIPIC source angles,* $(\theta_{CIPIC}, \phi_{CIPIC})$*, ear canal set-back angles,* (ϕ_R, ϕ_L)*, and the sphere radius R.*

```
io.whosmat('subject_200/hrir_final.mat')

[('hrir_l', (25, 50, 200), 'double'),
 ('hrir_r', (25, 50, 200), 'double')]
```

An example HRIR plot, similar to Figure 3, is shown in Figure 14.

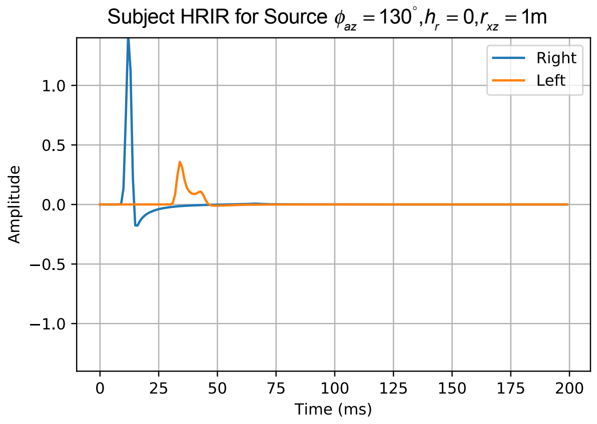


Fig. 14: Example right/left HRIR plots for a particular arrival angle pulled from the CIPIC-like database entry created for a radius 8.75 cm sphere.

Casual listening tests with a coarse fit human subject from CIPIC and the simple spherical model, indicate both similarities and differences. Coarse localization is similar between the two, but the spherical model seems *sterile*, that is the sound seems unnatural. The fact that coarse localization is present is an indication that the database is correct. Additional testing is planned.

Conclusions and Future Work

Development of the real-time signal processing aspect of the 3D audio simulator was a relatively simple task. This is a perfect application for the `pyaudio_helper` code module of `scikit-dsp-comm`. Working through the details of the coordinate transformations, and gain and parallax corrections on the geometry side, was a more complex undertaking. Likewise, working with the spherical head model calculations, first in the frequency domain (HRIR), and then the time domain (HRIR), was the most complex. The fact that recursions can be used to evaluate the needed special functions for sound pressure on the surface of a sphere, makes the generation of a CIPIC-like database entry take only a few minutes of compute time.

Informal testing of human subjects has gone well. Precise localization experiments using the static app have not been attempted just yet, as a formal pool human subjects has yet to be gathered. The virtual reality aspects of the dynamic app have received many positive comments from informal testing with those interested in 3D audio.

For future research, this simulator will be used to evaluate personalized HRIR fitting to human subjects, based on their upper torso anthropometrics. For the case of the spherical head, it is of interest to consider alternative HRIR grids. The 1 m radius 1250 point grid of angle pairs is no longer a limitation. For close range sound localization a different grid of angle pairs and with $r < 1$ m, can be used. This would make filter switching on the real-time DSP side of things finer grained, and hence more natural.

The Jupyter notebooks used in the analysis and development of this paper can be found on GitHub [3D_Audio]. This will give open access to anyone interested in trying out the simulator.

References

[HCI] *3-D Audio for Human/Computer Interaction*, (2019, June 30). Retrieved June 30, 2019, from https://www.ece.ucdavis.edu/cipic/spatial-sound/tutorial.

[CIPIC] *The CIPIC Interface Laboratory Home Page*, (2019, May 22). Retrieved May 22, 2019, from https://www.ece.ucdavis.edu/cipic.

[CIPICHRTF] *The CIPIC HRTF Database*, (2019, May 22). Retrieved May 22, 2019, from https://www.ece.ucdavis.edu/cipic/spatial-sound/hrtf-data.

[Fitzpatrick] Fitzpatrick, W., Wickert, M., and Semwal, S. (2013) 3D Sound Imaging with Head Tracking, *Proceedings IEEE 15th Digital Signal Processing Workshop/7th Signal Processing Education Workshop*.

[Wickert] *Real-Time Digital Signal Processing Using pyaudio_helper and the ipywidgets*, (2018, July 15). Retrieved May 22, 2019, from DOI 10.25080/Majora-4af1f417-00e.

[ScipySignal] *Signal processing (scipy.signal)*, (2019, May 22). Retrieved May 22, 2019, from https://docs.scipy.org/doc/scipy/reference/signal.html.

[KEMAR] GRAS Sound & Vibration A/S, Head & Torso Simulators, from http://www.gras.dk/products/head-torso-simulators-kemar.

[Beranek] Beranek, L. and Mellow, T (2012). *Acoustics: Sound Fields and Transducers*. London: Elsevier.

[Duda] Duda, R. and Martens, W. (1998). Range dependence of the response of a spherical head model, *J. Acoust. Soc. Am. 104 (5)*.

[Bogelein] Bogelein, S., Brinkmann, F., Ackermann, D., and Weinzierl, S. (2018). Localization Cues of a Spherical Head Model. *DAGA Conference 2018 Munich*.

[3D_Audio] 3D audio simulator, (2019, June 16): Retrieved June 16, 2019, from https://github.com/mwickert/3D_Audio_Simulator.