Student Name: Emil Sønderskov Hansen
Student ID: 20196042

Sound Processing
Mini-Project: Efficient HRTF filter for 3D sound

AALBORG UNIVERSITY

Intro

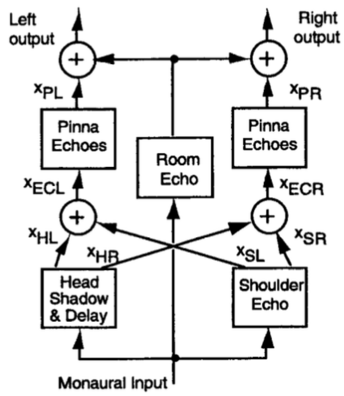

Figure 1: All of the components of the HRTF model

In the HRTF model proposed by Brown and Duda, various modules simulate different aspects of sound. The model's purpose is not to replicate physical processes but to create a simple system that can effectively convey impressions of spatial dimensions. The model takes a monaural input and processes it through a head model, followed by a pinna model, to produce azimuth and elevation effects. Additionally, the output from a room model is incorporated to generate range effects. In this project, a MatLab implementation of the head, pinna, and room models is presented and combined to create a complete HRTF model, as seen in figure 1. The paper found the shoulder model (from figure 1) to provide weak localization cues, which was therefore omitted from the paper and this implementation.

The Head Model

As seen in figure 2, the Head Model splits the signal to left and right. And each signal then goes through a Head Shadow filter (Interaural Level Difference (ILD)) and a delay (ITD).

Head Shadow filter

The Head Shadow filter is introduced in the paper as an analog one-pole/one-zero transfer function, as follows:

$$H(s,\theta) = \frac{\alpha(\theta)s + \beta}{s + \beta}, where \beta = \frac{2c}{a}$$

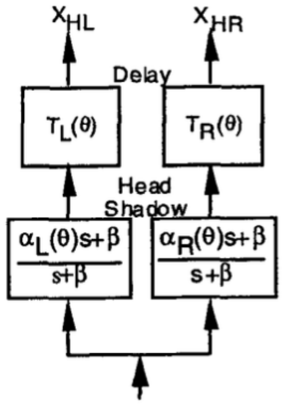

Figure 2: The components of the head model

The transfer function has a fixed pole at $-\beta$ and a zero depending on the azimuth value. The value of α is dependent on whether the input is the left or the right signal; $\alpha_L(\theta) = 1 - sin(\theta)$ and $\alpha_R(\theta) = 1 + sin(\theta)$.

The proposed transfer function is analog (i.e., a continuous time transfer function), and a digital version (i.e., discrete-time version) was approximated using a bilinear transform (also called the Tustin method). The bilinear transform states that the digital version can be derived by $s = \frac{2}{T}\frac{z-1}{z+1}$. The following is the approximated discrete-time transfer function.

$$H(z, \theta) = \frac{\alpha(\theta) + T\beta + z^{-1}(-2\alpha(\theta) + T\beta)}{2 + T\beta + z^{-1}(-2 + T\beta)}$$

And from the transfer function, the difference equation can be derived and is used in the Matlab Head Shadow function (see code below).

$$y(n) = \frac{(2\alpha(\theta) + T\beta)x(n) + (-2\alpha(\theta) + T\beta)x(n-1) - (-2 + T\beta)y(n-1)}{2 + T\beta}$$

This difference equation gave the desired results and contributed to useful localization cues.

It was also attempted to use the build-in Matlab function *c2d(system,T,'tustin')* followed by the function *impulse(system)* instead, to create an impulse response to convolute with the input signal. However, this did not give the desired result for unknown reasons and distorted the system heavily (this code is also attached).

The full derivations can be found in the appendix.

```
function[out] = HeadShadow(x, beta, alfa, T)

% creating an empty vector for output
y = zeros(length(x), 1);

% going through each sample and applying a head shadow filter
for n = 2:(length(x))
```

```
    y(n) = ((2*alfa+(T*beta))*x(n)+(-2*alfa+(T*beta))*x(n-1)-(-2+(T*beta))*y(n-1))/(2+(T*beta));
end

out = y;
```

Delay

The two delays are calculated based on the radius on the radius of the head (a), the azimuth (θ), and the speed of the sound (c). The article proposed the following calculations for delay for left and right and is implemented in the code:

$$T_L(\theta) = \frac{a + a\theta}{c}, T_R(\theta) = \frac{a - asin(\theta)}{c}$$

The delays are calculated and multiplied by the sampling rate of the input to find the number of samples to delay the signal with. The calculated was floored to an integer, providing the ITD cues good enough. Adding fractional delay did not seem to provide any further cues. The Matlab implementation can be seen below.

```
function[outL,outR] = ITD(xl,xr,fs,a,c, theta)

TlD = floor(((a + a*theta)/c)*fs);
TrD = floor(((a - a*sin(theta))/c)*fs);

% add delay (zeros) to beginning of signal depending on delay
yl = [zeros(TlD, 1); xl];
yr = [zeros(TrD, 1); xr];

% adding extra to the shorter signal so vectors have equal length
if(length(yl) > length(yr))
   diff = length(yl) - length(yr);
   yr = [yr; zeros(diff,1)];
else
   diff = length(yr) - length(yl);
   yl = [yl; zeros(diff,1)];
end

outL = yl;
outR = yr;
```

Pinna Echoes

While the Head Model provides cues for azimuth, the Pinna Model (see figure 3) provides elevation cues (as it is dependent on elevation (ϕ)). The pinna echoes are represented by five events (i.e., time delays) by τ_k, which each have an amplitude of ρ_k. The time delays for the five events are calculated with the following equation:

$$\tau_k(\theta, \phi) = A_k cos(\frac{\theta}{2}) sin(D_k(90^\circ - \phi)) + B_k$$

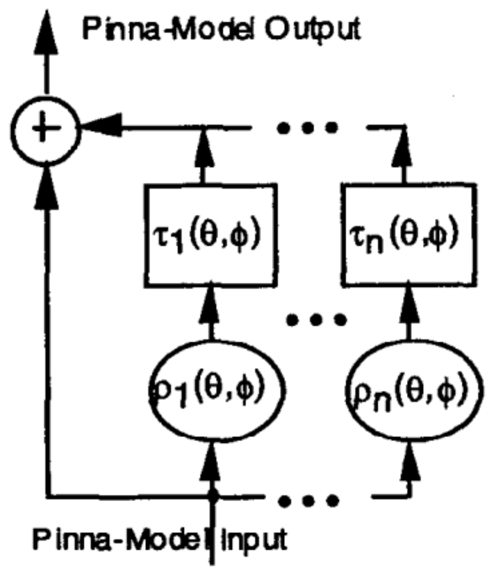

Figure 3: The model to represent the Pinna Echoes of five different events (i.e., time delays)

k	ρ_k	A_k	B_k	D_{k1}
1	.5	1	2	1
2	-1	5	4	.5
3	.5	5	7	.5
4	-.25	5	11	.5
5	.25	5	13	.5

Figure 4: Pinna Echo coefficients proposed by the paper

The article proposes coefficients used for calculating the delays. The coefficients are shown in figure 4.

A finite impulse response (FIR) was implemented to represent the pinna echoes. Firstly, all delays are calculated, and then an empty FIR with the number of taps of the highest delay is created. The first FIR tap is set to 1 as the input signal passes through (see figure 3). Each event is then placed in the impulse response based on the calculated delays with a value of ρ_k. In the implementation, it was tried to floor the delay to an integer. However, this did not seem to provide precise cues for elevation. The cues were found to be more accurate when distributing the value of ρ_k on two samples based on the fraction of the delay. Lastly, the input is convoluted with the impulse response. The elevation cues from the impulse response were not perceived as completely accurate. However, it must be considered that the article mentions D_k, to be adapted to the individual listener. The Matlab implementation can be seen below

```
function[out] =  PinnaEchoFunction(x,theta,phi)

P = [0.5 -1 0.5 -0.25 0.25]; % the amplitude of each echo
```

```

% all constants for the pinnae echo
A = [1 5 5 5 5];
B = [2 4 7 11 13];
D = [1 0.5 0.5 0.5 0.5];

%calculating all of the delays
d = zeros(5,1);
for n = 1:5
   d(n) = A(n)*cos(abs(theta)/2)*sin(D(n)*((90*(pi/180))) - phi) + B(n);
end

% creating a FIR based on the delays and the magnitude of each delay
% first impulse is set to 1, as the input signal also passes through
% (based on the article)

h = zeros(max(ceil(d)),1); % the impulse response as the length of the longest delay
h(1) = 1; % first to one (input signal passing through)
for n = 1:5
    % as we want the fractional part of the delay as-well, the ceil is
    % found which is d2, and then d1 as the sample before in the FIR.
    % we use the fractional part to calculate the weight of the two.
    d2 = ceil(d(n));
    d1 = d2 - 1;

    % calculating the fraction of the delay by subtracting the floored delay from
    % the delay
    frac = d(n) - floor(d(n));

    w2 = frac;     % the weight of d2 based on the fractional part
    w1 = (1-frac); % the weight of d1 based on the fractional part

    h(d2) = h(d2) + P(n) * w2;

    % as it is possible for d1 to be 0 (as it is subtracted by 1) we need
    % to check it is over 0
    if(d1 > 0)
        h(d1) = h(d1) + P(n) * w1;
    end
end

out = conv(x,h);
```

Room Model

The room model proposed in the paper is simply the original mono signal delayed and gained down (see figure 5). Firstly the signal is delayed by τ_E proposed to be 15 ms. And afterward gained down by K_E to be 15db lower than the channel gains K_L and K_R.

A gain for the channels was implemented, and the gain of the room model was fixed to be 15 dB lower. The code can be seen below.

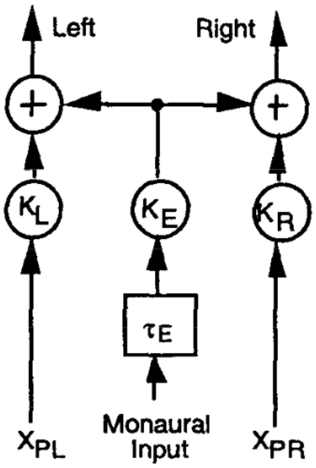

Figure 5: The Room Model

```
function[out] = RoomModel(x,yl,yr,gainDB,fs)

%creating 15 ms of delay in samples as suggested by the paper
d = floor(fs * 0.015);
xd = [zeros(d,1); x];

% if delayed signal is longer than the HRTF signal add the difference to
% make all signals same length
if(length(yl) > length(xd))
    diff = length(yl) - length(xd);
    xd = [xd; zeros(diff,1)];
else
    diff = length(xd) - length(yl);
    yl = [yl; zeros(diff,1)];
    yr = [yr; zeros(diff,1)];
end

% adding a amplitude (Kl and Kr) to the signal
A = db2mag(gainDB);
yl = A .* yl;
yr = A .* yr;

% making the room delay 15 db lower
xd = db2mag(gainDB - 15) .* xd;

% adding the delayed signal to left and right
y = [(yl + xd) (yr + xd)];

out = y;
```

1 The full HRTF code

All the different functions were gathered in one HRTF function as seen below:

```
function[out] = HRTF(x,fs,azimuthAngle,elevationAngle,headRadiusCM,gainDB)

% force to be mono if stereo as the paper specifies a mono signal as input
if(size(x,2) >= 2)
    x = sum(x, 2) / size(x, 2);
end

% setting variables needed
c = 343;
theta = azimuthAngle * (pi/180);
phi = elevationAngle * (pi/180);
a = headRadiusCM/100;
T = 1/fs;

% head shadow varables
beta = (2*c)/a;
alfaL = 1-sin(theta);
alfaR = 1+sin(theta);

% adding head shadow to singal
yl = HeadShadow(x,T,alfaL,beta);
yr = HeadShadow(x,T,alfaR,beta);

% add delay (zeros) to beginning of signal
[yl,yr] = ITD(yl,yr,fs,a,c,theta);

% PANNAE ECHO
yl = PinnaEchoFunction(yl,theta,phi);
yr = PinnaEchoFunction(yr,theta,phi);

% creating the "room model" / mix
y = RoomModel(x,yl,yr,gainDB,fs);

out = y;
```

2 Appendix

2.1 Head Shadow Model: Discrete-time transfer function

$$H(z,\theta) = \frac{\alpha(\theta)\left(\frac{2}{T}\frac{z-1}{z+1}\right)+\beta}{\left(\frac{2}{T}\frac{z-1}{z+1}\right)\ +\ \beta} = \frac{\frac{2\alpha(\theta)(z-1)}{T(z+1)}+\frac{T\beta(z+1)}{T(z+1)}}{\frac{2(z-1)}{T(z+1)}+\frac{T\beta(z+1)}{T(z+1)}} \ - \ getting\ common\ denominator$$

$$= \frac{\frac{2\alpha(\theta)(z-1)+T\beta(z+1)}{T(z+1)}}{\frac{2(z-1)+T\beta(z+1)}{T(z+1)}} = \frac{2\alpha(\theta)(z-1)+T\beta(z+1)}{2(z-1)+T\beta(z+1)} \ - \ getting\ rid\ of\ denominator$$

$$= \frac{z2\alpha(\theta)\left(1-z^{-1}\right)+zT\beta\left(1+z^{-1}\right)}{z2\left(1-z^{-1}\right)+zT\beta\left(1+z^{-1}\right)} = - \ using\ that\ zz^{-1}=z^0=1\ to\ get\ z^{-1}\ as\ part\ of\ eqaution$$

$$= \frac{2\alpha(\theta)\left(1-z^{-1}\right)+T\beta\left(1+z^{-1}\right)}{2\left(1-z^{-1}\right)+T\beta\left(1+z^{-1}\right)} = \frac{2\alpha(\theta)-z^{-1}2\alpha(\theta)+T\beta+z^{-1}T\beta}{2-2z^{-1}+T\beta+z^{-1}T\beta} \ - \ getting\ rid\ of\ z$$

$$= \frac{(2\alpha(\theta)+T\beta)+z^{-1}(-2\alpha(\theta)+T\beta)}{(2+T\beta)+z^{-1}(-2+T\beta)} \ - \ final\ discrete\ time\ transfer\ function$$

Figure 6: Billinear transform to approximate the discrete-time transfer function from the continuous-time transfer function

2.2 Head Shadow Model: Difference equation derivation

$$H(z,\theta) = \frac{2\alpha(\theta)+T\beta+z^{-1}(-2\alpha(\theta)+T\beta)}{2+T\beta+z^{-1}(-2+T\beta)} = \frac{Y(z)}{X(z)}$$

$$\Rightarrow Y(z)\Big(2+T\beta+z^{-1}(-2+T\beta)\Big) = X(z)\Big(2\alpha(\theta)+T\beta+z^{-1}(-2\alpha(\theta)+T\beta)\Big)$$

$$\Rightarrow (2+\mathrm{T}\beta)\mathrm{Y(n)} + \Big(-2 + \mathrm{T}\beta\Big)\mathrm{Y(n-1)} = (2\alpha(\theta)+T\beta)\mathrm{X(n)} + (-2\alpha(\theta)+T\beta)X(n-1)$$

$$\Rightarrow (2+\mathrm{T}\beta)\mathrm{Y(n)} = (2\alpha(\theta)+T\beta)\mathrm{X(n)} + (-2\alpha(\theta)+T\beta)X(n-1) - \Big(-2 + \mathrm{T}\beta\Big)\mathrm{Y(n-1)}$$

$$\Rightarrow \mathrm{Y(n)} = \frac{(2\alpha(\theta)+T\beta)\mathrm{X(n)} + (-2\alpha(\theta)+T\beta)X(n-1) - \Big(-2 + \mathrm{T}\beta\Big)\mathrm{Y(n-1)}}{(2+\mathrm{T}\beta)}$$

Figure 7: Difference equation derived from the approximated discrete-time transfer function