ATRテクニカルレポート表紙

[公開]

TR-M-0040

一次反射音付加による仮想空間の広さの制御

立蔵　洋介
Yosuke TATEKURA

西村　竜一
Ryouichi NISHIMURA

1998.9.25

ATR 知能映像通信研究所

実習報告書

一次反射音付加による仮想空間の広さの制御

実習先：ATR 知能映像通信研究所

実習指導者：西村 竜一

実習期間：平成 10 年 8 月 3 日～ 9 月 25 日

奈良先端科学技術大学院大学 情報科学研究科

立蔵 洋介

1　はじめに

実空間の状態認知において、聴覚は視覚に次いで多くの情報量を与える感覚である。特に後方など、人間の視野外の状況を認知するためには、音のもつ情報量は膨大であり、その寄与は大きい。

一方、情報通信技術の発達に伴い、マルチメディア時代に即した新しいコミュニケーションの手法の確立が急務となっている。その一翼として、目的に応じた仮想現実感の表現が期待されているが、これを実現する手段としては映像を三次元的に表示するのみならず、音を三次元的に表現する必要がある。これは前述したとおり、例えば前方の距離感など、視野内の状況を補足する情報源にとどまるだけではなく、視野外の状況認知の上でも重要な情報源だからだ。したがって、旧来の二次元的再生では音源との距離感や方向性などといった情報を十分に提供することができず、新たに任意の音響空間を再現する技術が求められる。

ところで室内の反射音は音に広がりや奥行き感を与えるため、臨場感を出す上で重要な役割を果たしている。人間が室内で聞く音には、音源から直接到達する直接音と、壁や天井、床に反射して到達する反射音とがある。1，2回ほど反射して数分の1秒後に届く初期反射どの方向からどのような時間遅れで届くかによって、人間はその室内の形状や大きさを知覚することができる。

そこで本研究では仮想空間の広さを聴覚だけで再現する方法として、音源から発せられた直接音と、壁などによる一次反射音をも考慮して提示することにより、これを実現する方法を試みた。ここで直接音の音像定位や反射音の到達方向をできるだけ正確に再現するため、直接音と反射音は、両耳間レベル差と両耳間位相差（時間差）に関する情報を内包した頭部伝達関数 (HRTF) を原音に畳み込むことによって生成した。

2　3次元音場再現の基礎技術

2.1　方向定位と頭部伝達関数

聴覚による方向定位は、両耳に到達する音の音圧差と位相差（時間差）によって実現されている。反射音がなくて音源から離れている場合、低い周波数では回折効果が小さいので音圧差は無視できるが、音源との距離による位相差が生じる。高い周波数では位相差が360°以上となって識別できなくなるが、頭、耳界などの回折効果による音圧差が生じる。このように低周波では主に位相差が、高周波では主に音圧差が方向定位に用いられている。

人間が音の方向定位を行うときにはこの特性を用いるだけではなく、頭部の形状や、耳の形状を基に判別している。これは頭の存在の影響で、到来方向によって微妙に変わる音質の違いを記憶しているのである。

したがって、後述するバイノーラル再生を用いて全方位の方向定位をも考慮した音を提示するためには原音に対して、両耳間レベル差と両耳間位相差（時間差）に関する情報を内包した頭部伝達関数 (HRTF) を畳み込んでやればよい。

頭部伝達関数は、一般に人間の頭部を模したダミーヘッドの両耳に仕込まれたマイクで、インパルス応答を計測されることによって得られる。技術的な問題点としては、いかに人間の頭部伝達特性を実現するかにある。なぜならば頭部や耳の形状は人間の固体差が非常に大きいからである。

2.2　バイノーラル再生

原音上における音圧を、ヘッドホンを利用して厳密に再生する方法として、バイノーラル再生が知られている。収音された音響信号は、ヘッドホンを用いて受聴者の耳元で再生される。この方法は受聴者の頭部回転・移動に対する対処の問題、ヘッドホン特性混入の問題などがあり、何よりも頭部にヘッドホンをかけるという、肉体的に制限が生じるという問題がある。しかし従来のスピーカによる再生とは違い、右耳の

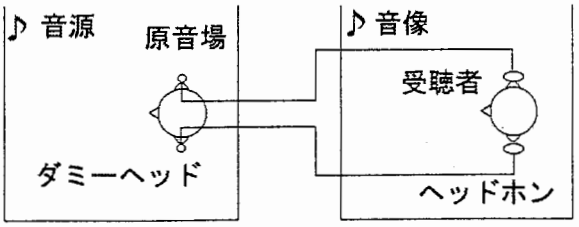

図 1: バイノーラルシステムの構成

みに聞こえる音が左耳にも聞こえてしまうといういわゆる空間のクロストークの現象を除去することができる。また現状では臨場感をもっとも良好に伝える再生方式の一つである。

3 反射音作成の汎用プログラムの開発

3.1 はじめに

後述する本実験で提示するための反射音を作成するための汎用プログラムを MATLAB 上で開発した。本節ではプログラムの内容、処理手順について述べる。

3.2 音源と観測者の座標による各音の方向と距離

観測点に対して、直接音と仮想空間に応じた反射音とを提示するためには、反射音と同等の性質を有する、反射音用の仮想音源の位置を割り出す必要がある。この音源の位置を割り出しには、図 2 のように壁を対称軸とした、鏡像法を用いればよい。この方法により、四方の壁と天井、床による計 6 つの反射音用仮想音源の位置を割り出した.

割り出した音源の位置より、観測点からみた各音の入射角、水平角、および伝達距離を求めた.

3.3 頭部伝達関数の補間

今回用いた頭部伝達関数は MIT の WWW 上で公開されているものである。この伝達関数は

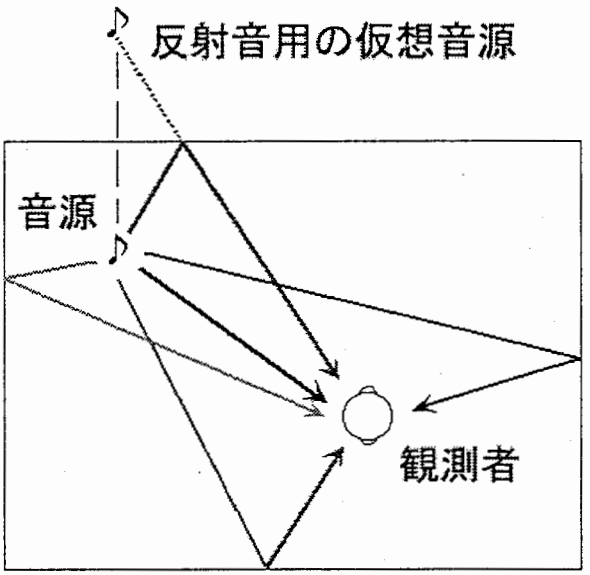

図 2: 仮想空間と伝達音のイメージ

ダミーヘッドの両耳で、ダミーヘッドから 1.4m 離れた位置に設置したスピーカから発せられたインパルスの応答を計測したものである。サンプリング周波数は 44.1KHz で、仰角-40 ° から 90 ° 内の 710 点で計測されている。

しかし仰角の分解能は 10 ° であり、各仰角における水平角の分解能も最小で 10 ° である。実際にはファイルで与えられている角度以外のデータが必要な場合の方が多い。そこで与えられた伝達関数より、新たに要求する伝達関数を求めることを試みた。

具体的な方法としては、求める伝達関数の近傍の伝達関数 2 つを FFT(高速フーリエ変換) によって時間領域から周波数領域に変換した。変換した 2 つの伝達関数を、求める伝達関数との角度の差に応じて比例配分して合成した。この手法は仰角成分での補正を行った後、水平成分の補正を行っている。

例として仰角 0 ° での条件下で、水平角 30 ° と 40 ° のデータから 36 ° のデータを求めたものを図 3 から図 5 に示す。これらのデータを見るに、合成した頭部伝達関数は与えられていた伝達関数の傾向を十分に捉えていると思われる。

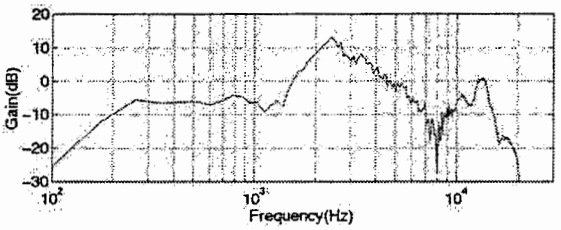

図 3: 水平角 30 ° の頭部伝達関数

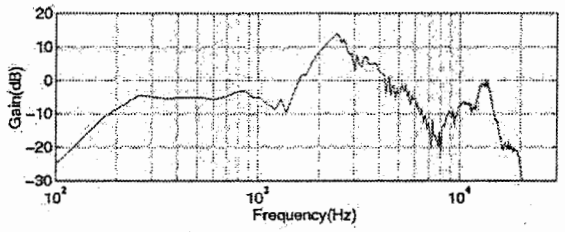

図 4: 水平角 40 ° の頭部伝達関数

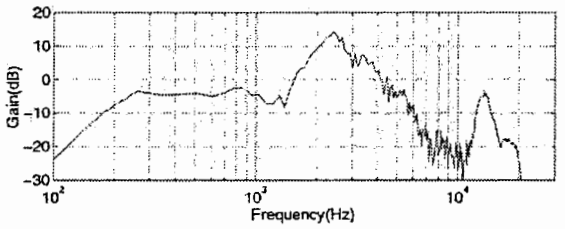

図 5: 合成して作成した水平角 36 ° の頭部伝達関数

3.4 距離減衰

音源から出た音は、媒質を伝播するにしたがって減衰する。これは媒質が音を吸収しているからである。よって自由空間に置かれた点音源から出た音の強さは、音源からの距離に反比例する。

そこで本プログラムでは、直接音の音源、および反射音仮想音源から観測点までの距離に応じた距離減衰を考慮して提示するために、頭部伝達関数を畳み込んだ音信号に対して、距離に相当するパワー減衰をかけた。

3.5 直接音と反射音の加算

最後にこれまでの作業で出来上がった直接音と 6 つの反射音を加算した。加算の際には、反射音の位相遅れを考慮した。すなわち各音源から観測点までの距離を求めた場合、当然ながら直接音の音源距離が最短である。したがって各反射音には距離の差に応じたゼロ点を信号の頭に埋め込むことによってこれを実現した。

4 実験内容

聴覚による空間の広さ知覚の検証のため、本プログラムで作成した音を用いた聴取実験を行った。

ピアノ演奏を基にした 5 種類の音の中から 2 種類を被験者にヘッドホンを介して提示し、どちらが広い空間で演奏されたように聞こえるかを 5 段階評価により判断させた。提示した 5 種類の音は

1. モノホニック (片耳だけに音を提示)
2. ダイオティック (両耳に同じ音を提示)
3. 直接音のみ提示
4. 直接音、および広い空間を想定した反射音を提示
5. 直接音、および狭い空間を想定した反射音を提示

である。

音源と観測点の配置については、真正面と真後ろにおける方向知覚の曖昧さを考慮して、観

測点の正面から右約 11° を想定した場所に音源を設置した。また 3. から 5. については、部屋の中心からの音源の位置、および観測点の位置を固定したままとした。5. の空間は 4. よりも狭い空間を想定している。

1. および 2. については頭部伝達関数を畳み込まず、モノラル録音された原音そのままを提示した。音の提示はどの場合でも約 10 秒で、音と音の間には約 4 秒の空き時間を置いた。

なお、5 種類の中から 2 種類を選んで提示するので組み合わせ数は 10 通りであるが、順序効果を考えて、一人の被験者につき 20 通りを提示した。

5 結果

被験者 5 人による実験結果をシェッフェの一対比較法を基に処理したものを図 6 に示す。数直線に示す数字は、尺度値 0.0 を示した刺激 (この場合モノホニックの音) に対する他の音の尺度の相対値である。尺度値差 1.0 で、被験者に判断してもらった 5 段階評価の 1 段階差分の半分の評価に相当する。また、各評価間の差の有無を検定するため、ヤードスティックによる検定を行った。

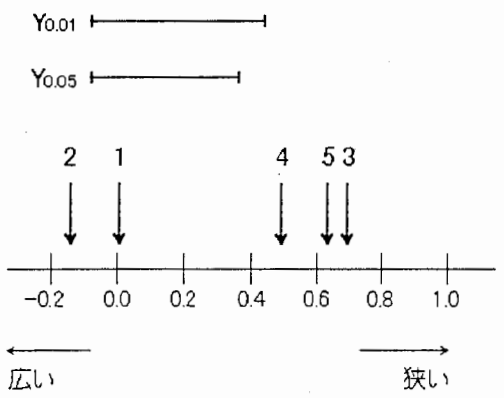

図 6: 聴取実験結果

その結果、頭部伝達関数の畳み込みの有無によって、結果が二極化した。すなわち、頭部伝達関数を畳み込まなかった方がおしなべて広い空間と知覚された。また、ヤードスティックによる検定では危険率 5 %、1 %共に、おおよそこの 2 つのグループの差は有意であるものの、グループ内では有意な差はどちらも得られていない。すなわち、直接音のみを提示した場合と、反射音をも提示した場合とで、有意な差は得られなかった。

6 考察

本実験がこのような結果を示したことについて、考えられる原因は大別して 2 通り、すなわち物理的原因と心理的原因があると考えられる。

まず物理的原因について考える。これは距離減衰における周波数特性を考慮しなかったためであると思われる。すなわち、距離減衰の原因として媒質による音波の吸収を挙げたわけだが、もう少し物理的に考えると、これは媒質の粘性による粒子運動のエネルギー減衰に起因する。吸収の程度は温度、湿度の上昇と共に小さくなるわけだが、周波数にも依存し、高い周波数では急速に増大する。よって空間内での距離減衰をより正確に表現するためには、低周波の音ほど距離減衰の割合を小さくし、高周波の音ほど距離減衰を大きくする必要がある。しかし本プログラムではその点を考慮せず、時間領域上でパワーに減衰をかけたのみなので、空間の伝達特性を正確に再現していない。頭部伝達関数は、耳界から鼓膜までの伝達特性を考慮しているのみであり、音源から耳界までの伝達特性を考慮したものではない。今回、頭部伝達関数を畳み込んだ音に関して、総じて中、高周波の音が強調され、低周波の音が弱く聞こえたのはそのためであると思われる。上述の周波数特性の様子を図 7 から図 9 に示す。

次に心理的原因であるが、こちらは反射音に対する先験的知覚によるものではないかと考えられる。ある被験者に実験後、話を聞いたところ、反射音を多く含んだ音ほど広い空間に知覚したという意見があった。これは実験者の意図に反するものであるが、なぜ被験者がそのように知覚したのかを分析する。今回提示した音は

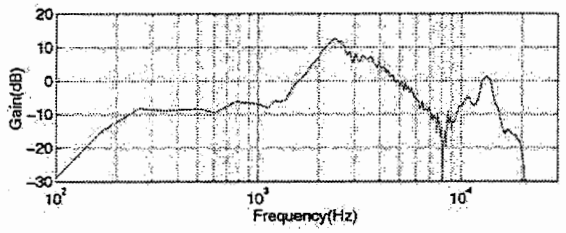

図 7: 水平角 11 ° の頭部伝達関数

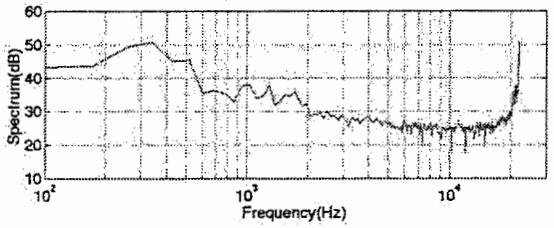

図 8: 提示した音のスペクトル (HRTF 畳み込み前)

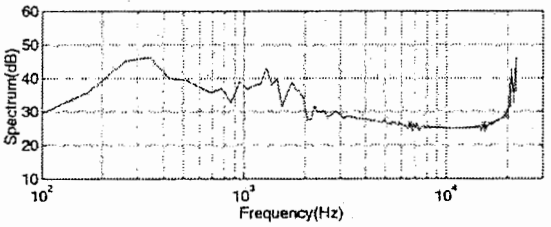

図 9: 提示した音のスペクトル (HRTF 畳み込み後)

ピアノの演奏音であるが、ピアノの演奏経験などのない限り、たいていの人間の場合ピアノの実音に接するのは小学生の頃の経験などより、音楽室と体育館といったごく限定された場合であることが多い。この二つの空間を比較すると、空間の大きさは明らかに体育館のほうが大きい。そして残響音も両者の構成材質を比較すると、体育館のほうが残響が豊富である。これらの経験から、残響を多く含んだ場合ほど、空間が広いと知覚する場合があるのではないかと思われる。

7 結論

反射音により空間の広さを制御する方法の検討として、聴覚のみに刺激を与えて空間の広さを知覚させる被験者実験を行った。その結果、直接音のみの提示と反射音を付加した提示で、有意な差は認められなかった。また頭部伝達関数を畳み込まないほうが広く感じられるという結果が得られた。これを回避には、単にパワーを減衰させる距離減衰ではなく、空間伝達の周波数特性を含んだ距離減衰をかけて、提示する音の周波数応答の調整を行うという方法を導入する必要がある。

また、今回はピアノの演奏音のみによる提示であったが、これとは異なる音を提示し、同様の実験を行い、比較する必要があると思われる。あるいは、音だけではなく、映像と同時に提示した場合を調査する必要もある。

謝辞

本報告書の執筆を終えるに際し、今回実習の機会を与えて下さいました ATR 知能映像通信研究所第 5 研究室の宮里勉室長に深甚の感謝を表すると共に、終始本研究のご指導いただきました同研究室の西村竜一氏に心より謝意を表します。また日頃より多大なるご協力をいただいた同研究室の諸兄をはじめとする、知能映像通信研究所の皆様に感謝いたします。

開発したプログラムソース

プログラムの仕様について

本研究で開発したMATLABプログラム mwave および nmwave のソース、およびその仕様などについて述べる。

今回開発した元のプログラムは mwave である。これは空間の大きさ、音源の位置、座標を入力すると、各音に相当したHRTFを読み込み（場合によっては合成し）、元の波形データファイルと畳み込んで提示するプログラムである。

しかしこのプログラムは作業をすべてRAM内で処理してしまうため、ファイルサイズが数MB以上の波形データを処理するのは不可能である。そこでHDを一時記憶として用い、理論上どの大きさのファイルも扱えるようにしたのが nmwave である。操作方法は両者とも共通であり、使用する外部関数も同じなので、以下に使用法を簡単に述べる。

空間の設定

空間の形状は直方体である。空間の大きさについては、xy平面内の第1像現を想定して作成した。従って、入力する値は正数が望まれる。

音源、観測点の設定

空間の設定と同様、正数が望まれる。なお、空間外に設定した場合についてはテストを行なっていないので、勧めない。

頭の角度の設定

頭の角度を設定しなかった（両方とも0にした場合)、デフォルトでは正面方向は、x軸の正の方向を向いている。水平角方向の成分に正の値を加えると、反時計回りの方向に正面方向を変えていく。

また仰角は０°である。

HRTFの水平角

HRTFの水平角成分は時計回りで３６０°回っている。すなわち、頭の角度設定とは回転方向が反対であるので、注意が必要である。

壁の反射率

壁等の反射率は変数 a2* で設定できる。 * は各反射音の番号を表している。

その他

パス設定は一部の外部関数内にもあるので、注意すること。

mwave.m のプログラムソース

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                              %
% プログラム ID : mwave.m                                      %
% 内容          : 部屋の大きさ、観測点、音源の座標から         %
%                 直接音と反射音を求める                       %
% 製作者        : yosuke-t                                     %
% 最終更新日    : 98.10.7                                      %
%                                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
%   各条件変数の初期化
%


%   座標系の初期設定
%
spo = [    5     8     1];              % 点音源の座標
obs = [   10     7   1.2];              % 観測点の座標
room = [ 16    12     3];               % 空間の大きさ
head = [   0   180];                    % 頭の角度 [elev azim]

%   壁の反射率の設定
%
a21 = 0.93;
a22 = 0.93;
a23 = 0.93;
a24 = 0.93;
a25 = 0.90;
a26 = 0.90;

%   元となる波形データの場所と名前
%
readwave = '/home/yosuke/wave/original/sss.dat';

%   直接音のみを出力する場所と名前
%
wwavedil = '/home/yosuke/wave/p_dl20.dat';
wwavedir = '/home/yosuke/wave/p_dr20.dat';
```

```
%  直接音＋反射音を出力する場所と名前
%
wwaverel = '/home/yosuke/wave/p_cl20.dat';
wwaverer = '/home/yosuke/wave/p_cr20.dat';


%
%  元となる波形データを読み込む
%
fid = fopen(readwave, 'r');
o_wave = fread( fid, 'short');
fclose(fid);


%
%  各音の到達方向、到達距離を求める
%

%  外部関数 drpt.m を用いて r phi theta を求める
%
rpt = drpt(spo, obs, room);

%  入力された頭の角度 head[elev azim] に応じて角度を補正する
%
for N=1:7

   rpt(N, 2) = rpt(N, 2) - head(1);     %  仰角成分の補正

   rpt(N, 3) = rpt(N, 3) + head(2);     %  水平角成分の補正
   if (rpt(N, 3) >= 360)
      rpt(N, 3) = rpt(N, 3) - 360;      %  水平角が３６０度を越えれば
   end                                  %  ３６０度引く

end


%
%  反射音の中から、到達距離が最大のものの値を求める
%
maxr = max( rpt );                  %  rpt から各成分の最大値を取り出す
maxd = dtime(maxr(1));              %  r の最大値分のゼロ成分を計算
```

```
%
%  外部関数 imp(elev, azim) により、必要な HRTF を読み込む
%        1:512 -> L   513:1024 -> R
%
imp1 = imp( rpt(1,2), rpt(1,3) );
imp21 = imp( rpt(2,2), rpt(2,3) );
imp22 = imp( rpt(3,2), rpt(3,3) );
imp23 = imp( rpt(4,2), rpt(4,3) );
imp24 = imp( rpt(5,2), rpt(5,3) );
imp25 = imp( rpt(6,2), rpt(6,3) );
imp26 = imp( rpt(7,2), rpt(7,3) );



%
%  HRTF と元の波形を畳み込む
%  外部関数 dd(wavedata, r) によって距離減衰をかける
%

%  直接音
%
wav1l = conv( o_wave, imp1(1:512) );
wav1r = conv( o_wave, imp1(513:1024) );
wav1l = dd( wav1l, rpt(1,1) );
wav1r = dd( wav1r, rpt(1,1) );

%  反射音 1
%
wav21l = conv( o_wave, imp21(1:512) );
wav21r = conv( o_wave, imp21(513:1024) );
wav21l = dd( wav21l, rpt(2,1) ) *a21;
wav21r = dd( wav21r, rpt(2,1) ) *a21;

%  反射音 2
%
wav22l = conv( o_wave, imp22(1:512) );
wav22r = conv( o_wave, imp22(513:1024) );
wav22l = dd( wav22l, rpt(3,1) ) *a22;
wav22r = dd( wav22r, rpt(3,1) ) *a22;

%  反射音 3
%
```

```
wav23l = conv( o_wave, imp23(1:512) );
wav23r = conv( o_wave, imp23(513:1024) );
wav23l = dd( wav23l, rpt(4,1) ) *a23;
wav23r = dd( wav23r, rpt(4,1) ) *a23;

%  反射音4
%
wav24l = conv( o_wave, imp24(1:512) );
wav24r = conv( o_wave, imp24(513:1024) );
wav24l = dd( wav24l, rpt(5,1) ) *a24;
wav24r = dd( wav24r, rpt(5,1) ) *a24;

%  反射音5
%
wav25l = conv( o_wave, imp25(1:512) );
wav25r = conv( o_wave, imp25(513:1024) );
wav25l = dd( wav25l, rpt(6,1) ) *a25;
wav25r = dd( wav25r, rpt(6,1) ) *a25;

%  反射音6
%
wav26l = conv( o_wave, imp26(1:512) );
wav26r = conv( o_wave, imp26(513:1024) );
wav26l = dd( wav26l, rpt(7,1) ) *a26;
wav26r = dd( wav26r, rpt(7,1) ) *a26;



%
%  外部関数 dtime(r) により各音の前後にゼロを埋め込む
%
wav1l = [ wav1l ; zeros(maxd, 1) ];
wav1r = [ wav1r ; zeros(maxd, 1) ];

wav21l = [ zeros(dtime(rpt(2, 1)), 1) ; wav21l ; zeros(maxd - dtime(rpt(2, 1)), 1) ];
wav21r = [ zeros(dtime(rpt(2, 1)), 1) ; wav21r ; zeros(maxd - dtime(rpt(2, 1)), 1) ];

wav22l = [ zeros(dtime(rpt(3, 1)), 1) ; wav22l ; zeros(maxd - dtime(rpt(3, 1)), 1) ];
wav22r = [ zeros(dtime(rpt(3, 1)), 1) ; wav22r ; zeros(maxd - dtime(rpt(3, 1)), 1) ];

wav23l = [ zeros(dtime(rpt(4, 1)), 1) ; wav23l ; zeros(maxd - dtime(rpt(4, 1)), 1) ];
wav23r = [ zeros(dtime(rpt(4, 1)), 1) ; wav23r ; zeros(maxd - dtime(rpt(4, 1)), 1) ];
```

```
wav24l = [ zeros(dtime(rpt(5, 1)), 1) ; wav24l ; zeros(maxd - dtime(rpt(5, 1)), 1) ];
wav24r = [ zeros(dtime(rpt(5, 1)), 1) ; wav24r ; zeros(maxd - dtime(rpt(5, 1)), 1) ];

wav25l = [ zeros(dtime(rpt(6, 1)), 1) ; wav25l ; zeros(maxd - dtime(rpt(6, 1)), 1) ];
wav25r = [ zeros(dtime(rpt(6, 1)), 1) ; wav25r ; zeros(maxd - dtime(rpt(6, 1)), 1) ];

wav26l = [ zeros(dtime(rpt(7, 1)), 1) ; wav26l ; zeros(maxd - dtime(rpt(7, 1)), 1) ];
wav26r = [ zeros(dtime(rpt(7, 1)), 1) ; wav26r ; zeros(maxd - dtime(rpt(7, 1)), 1) ];



%
%  直接音と各反射音を加算合成
%
p_wavel = ( wav1l + wav21l + wav22l + wav23l + wav24l + wav25l + wav26l );
p_waver = ( wav1r + wav21r + wav22r + wav23r + wav24r + wav25r + wav26r );



%
%  直接音をファイルに保存
%
fid = fopen( wwavedil, 'w' );
fwrite( fid,wav1l, 'short');
fclose( fid );
fid = fopen( wwavedir, 'w' );
fwrite( fid, wav1r, 'short');
fclose( fid );



%
%  直接音＋反射音をファイルに保存
%
fid = fopen( wwaverel, 'w' );
fwrite( fid,p_wavel, 'short');
fclose( fid );
fid = fopen( wwaverer, 'w' );
fwrite( fid, p_waver, 'short');
fclose( fid );

%%%%% End of file %%%%%
```

0.1 nmwave.m のプログラムソース

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                       %
% プログラム ID：nmwave.m                               %
% 内容         ：部屋の大きさ、観測点、音源の座標から   %
%                直接音と反射音を求める                 %
% 製作者       ：yosuke-t                               %
% 最終更新日   ：98.10.7                                %
% 備考         ：mwave.m のファイルの大きさ制限を除去   %
%                大型の波形ファイルの加工に対応         %
%                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
% 各条件変数の初期化
%

% 座標系の初期設定
%
spo  = [  1    5    1];               % 点音源の座標
obs  = [  6    5  1.2];               % 観測点の座標
room = [ 8    6    3];                % 空間の大きさ
head = [  0  180];                    % 頭の角度 [elev azim]

% 壁の反射率の設定
%
a21 = 0.90;
a22 = 0.90;
a23 = 0.90;
a24 = 0.90;
a25 = 0.60;
a26 = 0.70;

% 元となる波形データの場所と名前
%
readwave =  '/home/yosuke/wave/original/debussy.dat';

% 直接音のみを出力する場所と名前
%
wwavedl = '/home/yosuke/wave/flutedl.dat';
```

```
wwavedr = '/home/yosuke/wave/flutedr.dat';

%  直接音＋反射音を出力する場所と名前
%
wwavecl = '/home/yosuke/wave/flutecl.dat';
wwavecr = '/home/yosuke/wave/flutecr.dat';

%  処理中のデータを一時記憶するディレクトリ
%
temparea = '/home/tmp/yosuke';



%
%  各音の到達方向、距離を求める
%

%  外部関数 drpt.m を用いて r phi theta を求める
%
rpt = drpt(spo, obs, room);

%  頭の角度 head[elev azim] に応じて角度を補正する
%
for N=1:7

   rpt(N, 2) = rpt(N, 2) - head(1);    %  行角成分の補正

   rpt(N, 3) = rpt(N, 3) + head(2);    %  水平角成分の補正
   if (rpt(N, 3) >= 360)
      rpt(N, 3) = rpt(N, 3) - 360;     %  水平角が３６０度を越えれば
   end                                 %  ３６０引く

end

%
%  反射音の中から、到達距離が最大のものの値を求める
%
maxr = max( rpt );        %  rpt から各成分の最大値を取り出す
maxd = dtime(maxr(1));    %  r の最大値分のゼロ成分を外部関数 dtime.m で計算



%
```

```
%  外部関数 imp(elev, azim) を用いて、必要な HRTF を求める
%        1:512 -> L   513:1024 -> R
%
imp1 = imp( rpt(1,2), rpt(1,3) );        % source(a, b, c)
imp21 = imp( rpt(2,2), rpt(2,3) );       % source(-a, b, c)
imp22 = imp( rpt(3,2), rpt(3,3) );       % source(2l-a, b, c)
imp23 = imp( rpt(4,2), rpt(4,3) );       % source(a, -b, c)
imp24 = imp( rpt(5,2), rpt(5,3) );       % source(a, 2m-b, c)
imp25 = imp( rpt(6,2), rpt(6,3) );       % source(a, b, -c)
imp26 = imp( rpt(7,2), rpt(7,3) );       % source(a, b, 2n-c)



%
%  各音のデータの初期設定
%

%  反射音の波形の頭にゼロを埋め込む
%
wav1l = [ ]; wav1r = [ ];
wav21l = [zeros(dtime(rpt(2, 1)), 1)]; wav21r = [zeros(dtime(rpt(2, 1)), 1)];
wav22l = [zeros(dtime(rpt(3, 1)), 1)]; wav22r = [zeros(dtime(rpt(3, 1)), 1)];
wav23l = [zeros(dtime(rpt(4, 1)), 1)]; wav23r = [zeros(dtime(rpt(4, 1)), 1)];
wav24l = [zeros(dtime(rpt(5, 1)), 1)]; wav24r = [zeros(dtime(rpt(5, 1)), 1)];
wav25l = [zeros(dtime(rpt(6, 1)), 1)]; wav25r = [zeros(dtime(rpt(6, 1)), 1)];
wav26l = [zeros(dtime(rpt(7, 1)), 1)]; wav26r = [zeros(dtime(rpt(7, 1)), 1)];

%
%
wavbag1l = [zeros(511, 1)]; wavbag1r = [zeros(511, 1)];
wavbag21l = [zeros(511, 1)]; wavbag21r = [zeros(511, 1)];
wavbag22l = [zeros(511, 1)]; wavbag22r = [zeros(511, 1)];
wavbag23l = [zeros(511, 1)]; wavbag23r = [zeros(511, 1)];
wavbag24l = [zeros(511, 1)]; wavbag24r = [zeros(511, 1)];
wavbag25l = [zeros(511, 1)]; wavbag25r = [zeros(511, 1)];
wavbag26l = [zeros(511, 1)]; wavbag26r = [zeros(511, 1)];



%
%
%  元の波形データを所定の単位時間毎に区切り、各音に HRTF を畳み込む
%
```

```
%

% ディレクトリを一時記憶ディレクトリに変更
%
cd(temparea);

fw1lid = fopen('wav1l.tmp', 'w'); fwrite(fw1lid, wav1l, 'short');
fw1rid = fopen('wav1r.tmp', 'w'); fwrite(fw1rid, wav1r, 'short');

fw21lid = fopen('wav21l.tmp', 'w'); fwrite(fw21lid, wav21l, 'short');
fw21rid = fopen('wav21r.tmp', 'w'); fwrite(fw21rid, wav21r, 'short');

fw22lid = fopen('wav22l.tmp', 'w'); fwrite(fw22lid, wav22l, 'short');
fw22rid = fopen('wav22r.tmp', 'w'); fwrite(fw22rid, wav22r, 'short');

fw23lid = fopen('wav23l.tmp', 'w'); fwrite(fw23lid, wav23l, 'short');
fw23rid = fopen('wav23r.tmp', 'w'); fwrite(fw23rid, wav23r, 'short');

fw24lid = fopen('wav24l.tmp', 'w'); fwrite(fw24lid, wav24l, 'short');
fw24rid = fopen('wav24r.tmp', 'w'); fwrite(fw24rid, wav24r, 'short');

fw25lid = fopen('wav25l.tmp', 'w'); fwrite(fw25lid, wav25r, 'short');
fw25rid = fopen('wav25r.tmp', 'w'); fwrite(fw25rid, wav25r, 'short');

fw26lid = fopen('wav26l.tmp', 'w'); fwrite(fw26lid, wav26r, 'short');
fw26rid = fopen('wav26r.tmp', 'w'); fwrite(fw26rid, wav26r, 'short');

% 不要なデータをメモリから消去する
%
clear wav1l wav21l wav22l wav23l wav24l wav25l wav26l;
clear wav1r wav21r wav22r wav23r wav24r wav25r wav26r;



%
% 元の波形データのサイズをチェック
%
fid = fopen(readwave, 'r');
fseek(fid, 0, 1);
endfile = ftell(fid);
fseek(fid, 0, -1);
```

```
while ftell(flid) < endfile

  %
  %  元の波形データから2秒分読み込む
  %
  tmp = fread(fid, 88200, 'short');

  %
  %  読み込んだ波形データとHRTFを畳み込む
  %  外部関数dd(wave, r)によって距離減衰をかける
  %

  %  読み込んだ波形データとHRTFを畳み込んだ波形の信号数を求める
  %
  delta = size(tmp) + 512 - 1;

  %  直接音
  %
  wavtmp1l = conv( tmp, imp1(1:512) );
  wavtmp1r = conv( tmp, imp1(513:1024) );
  wavtmp1l = dd( wavtmp1l, rpt(1,1) );
  wavtmp1r = dd( wavtmp1r, rpt(1,1) );
  wavtmp1l( 1 : 511 ) = wavtmp1l( 1 : 511 ) + wavbag1l;
  wavtmp1r( 1 : 511 ) = wavtmp1r( 1 : 511 ) + wavbag1r;
  fwrite(fw1lid, wavtmp1l(1:size(tmpl)), 'short');
  fwrite(fw1rid, wavtmp1r(1:size(tmpr)), 'short');

  wavbag1l = wavtmp1l( (delta - 510): delta);
  wavbag1r = wavtmp1r( (delta - 510): delta);

  %  反射音1
  %
  wavtmp21l = conv( tmp, imp21(1:512) );
  wavtmp21r = conv( tmp, imp21(513:1024) );
  wavtmp21l = dd( wavtmp21l, rpt(2,1) ) * a21;
  wavtmp21r = dd( wavtmp21r, rpt(2,1) ) * a21;
  wavtmp21l( 1 : 511 ) = wavtmp21l( 1 : 511 ) + wavbag21l;
  wavtmp21r( 1 : 511 ) = wavtmp21r( 1 : 511 ) + wavbag21r;
  fwrite(fw21lid, wavtmp21l(1:size(tmpl)), 'short');
  fwrite(fw21rid, wavtmp21r(1:size(tmpr)), 'short');

  wavbag21l = wavtmp21l( (delta - 510): delta);
```

```
wavbag21r = wavtmp21r( (delta - 510): delta);

% 反射音 2
%
wavtmp22l = conv( tmp, imp22(1:512) );
wavtmp22r = conv( tmp, imp22(513:1024) );
wavtmp22l = dd( wavtmp22l, rpt(3,1) ) * a22;
wavtmp22r = dd( wavtmp22r, rpt(3,1) ) * a22;
wavtmp22l( 1 : 511 ) = wavtmp22l( 1 : 511 ) + wavbag22l;
wavtmp22r( 1 : 511 ) = wavtmp22r( 1 : 511 ) + wavbag22r;
fwrite(fw22lid, wavtmp22l(1:size(tmpl)), 'short');
fwrite(fw22rid, wavtmp22r(1:size(tmpr)), 'short');

wavbag22l = wavtmp22l( (delta - 510): delta);
wavbag22r = wavtmp22r( (delta - 510): delta);

% 反射音 3
%
wavtmp23l = conv( tmp, imp23(1:512) );
wavtmp23r = conv( tmp, imp23(513:1024) );
wavtmp23l = dd( wavtmp23l, rpt(4,1) ) * a23;
wavtmp23r = dd( wavtmp23r, rpt(4,1) ) * a23;
wavtmp23l( 1 : 511 ) = wavtmp23l( 1 : 511 ) + wavbag23l;
wavtmp23r( 1 : 511 ) = wavtmp23r( 1 : 511 ) + wavbag23r;
fwrite(fw23lid, wavtmp23l(1:size(tmpl)), 'short');
fwrite(fw23rid, wavtmp23r(1:size(tmpr)), 'short');

wavbag23l = wavtmp23l( (delta - 510): delta);
wavbag23r = wavtmp23r( (delta - 510): delta);

% 反射音 4
%
wavtmp24l = conv( tmp, imp24(1:512) );
wavtmp24r = conv( tmp, imp24(513:1024) );
wavtmp24l = dd( wavtmp24l, rpt(5,1) ) * a24;
wavtmp24r = dd( wavtmp24r, rpt(5,1) ) * a24;
wavtmp24l( 1 : 511 ) = wavtmp24l( 1 : 511 ) + wavbag24l;
wavtmp24r( 1 : 511 ) = wavtmp24r( 1 : 511 ) + wavbag24r;
fwrite(fw24lid, wavtmp24l(1:size(tmpl)), 'short');
fwrite(fw24rid, wavtmp24r(1:size(tmpr)), 'short');

wavbag24l = wavtmp24l( (delta - 510): delta);
wavbag24r = wavtmp24r( (delta - 510): delta);
```

```
  %  反射音5
  %
  wavtmp25l = conv( tmp, imp25(1:512) );
  wavtmp25r = conv( tmp, imp25(513:1024) );
  wavtmp25l = dd( wavtmp25l, rpt(6,1) ) * a25;
  wavtmp25r = dd( wavtmp25r, rpt(6,1) ) * a25;
  wavtmp25l( 1 : 511 ) = wavtmp25l( 1 : 511 ) + wavbag25l;
  wavtmp25r( 1 : 511 ) = wavtmp25r( 1 : 511 ) + wavbag25r;
  fwrite(fw25lid, wavtmp25l(1:size(tmpl)), 'short');
  fwrite(fw25rid, wavtmp25r(1:size(tmpr)), 'short');

  wavbag25l = wavtmp25l( (delta - 510): delta);
  wavbag25r = wavtmp25r( (delta - 510): delta);

  %  反射音6
  %
  wavtmp26l = conv( tmp, imp26(1:512) );
  wavtmp26r = conv( tmp, imp26(513:1024) );
  wavtmp26l = dd( wavtmp26l, rpt(7,1) ) * a26;
  wavtmp26r = dd( wavtmp26r, rpt(7,1) ) * a26;
  wavtmp26l( 1 : 511 ) = wavtmp26l( 1 : 511 ) + wavbag26l;
  wavtmp26r( 1 : 511 ) = wavtmp26r( 1 : 511 ) + wavbag26r;
  fwrite(fw26lid, wavtmp26l(1:size(tmpl)), 'short');
  fwrite(fw26rid, wavtmp26r(1:size(tmpr)), 'short');

  wavbag26l = wavtmp26l( (delta - 510): delta);
  wavbag26r = wavtmp26r( (delta - 510): delta);


end

%
%  元の波形データファイルのファイルポインタを閉じる
%
fclose(fid);


%
%  各音データファイルの終りにゼロを埋め込む
%
fwrite(fw1lid, [wavbag1l ; zeros(maxd, 1)], 'short');
fwrite(fw1rid, [wavbag1r ; zeros(maxd, 1)], 'short');
```

```
fwrite(fw21lid, [wavbag21l ; zeros(maxd - dtime(rpt(2, 1)),1)], 'short');
fwrite(fw21rid, [wavbag21r ; zeros(maxd - dtime(rpt(2, 1)),1)], 'short');

fwrite(fw22lid, [wavbag22l ; zeros(maxd - dtime(rpt(3, 1)),1)], 'short');
fwrite(fw22rid, [wavbag22r ; zeros(maxd - dtime(rpt(3, 1)),1)], 'short');

fwrite(fw23lid, [wavbag23l ; zeros(maxd - dtime(rpt(4, 1)),1)], 'short');
fwrite(fw23rid, [wavbag23r ; zeros(maxd - dtime(rpt(4, 1)),1)], 'short');

fwrite(fw24lid, [wavbag24l ; zeros(maxd - dtime(rpt(5, 1)),1)], 'short');
fwrite(fw24rid, [wavbag24r ; zeros(maxd - dtime(rpt(5, 1)),1)], 'short');

fwrite(fw25lid, [wavbag25l ; zeros(maxd - dtime(rpt(6, 1)),1)], 'short');
fwrite(fw25rid, [wavbag25r ; zeros(maxd - dtime(rpt(6, 1)),1)], 'short');

fwrite(fw26lid, [wavbag26l ; zeros(maxd - dtime(rpt(7, 1)),1)], 'short');
fwrite(fw26rid, [wavbag26r ; zeros(maxd - dtime(rpt(7, 1)),1)], 'short');


%
%  ファイルポインタをリセット
%

fclose('all');

fw1lid = fopen('wav1l.tmp', 'r'); fw1rid = fopen('wav1r.tmp', 'r');
fw21lid = fopen('wav21l.tmp', 'r'); fw21rid = fopen('wav21r.tmp', 'r');
fw22lid = fopen('wav22l.tmp', 'r'); fw22rid = fopen('wav22r.tmp', 'r');
fw23lid = fopen('wav23l.tmp', 'r'); fw23rid = fopen('wav23r.tmp', 'r');
fw24lid = fopen('wav24l.tmp', 'r'); fw24rid = fopen('wav24r.tmp', 'r');
fw25lid = fopen('wav25l.tmp', 'r'); fw25rid = fopen('wav25r.tmp', 'r');
fw26lid = fopen('wav26l.tmp', 'r'); fw26rid = fopen('wav26r.tmp', 'r');



fseek(fw1lid, 0, -1); fseek(fw1rid, 0, -1);
fseek(fw21lid, 0, -1); fseek(fw21rid, 0, -1);
fseek(fw22lid, 0, -1); fseek(fw22rid, 0, -1);
fseek(fw23lid, 0, -1); fseek(fw23rid, 0, -1);
fseek(fw24lid, 0, -1); fseek(fw24rid, 0, -1);
fseek(fw25lid, 0, -1); fseek(fw25rid, 0, -1);
fseek(fw26lid, 0, -1); fseek(fw26rid, 0, -1);
```

```
%
%  処理したファイルの大きさを調べる
%
fseek(fw1lid, 0, 1);
endfile2 = ftell(fw1lid);
fseek(fw1lid, 0, -1);


%
%  ファイルとして保存するデータの書き込み準備をする
%
fdlid = fopen( wwavedl, 'w');
fdrid = fopen( wwavedr, 'w');
fclid = fopen( wwavecl, 'w');
fcrid = fopen( wwavecr, 'w');


%
%  各ファイルを２秒分ずつ読み込んで足し合わせる
%
while ftell(fw1lid) < endfile2

  %  ２秒分の信号数
  %
  m = 88200;

  %  各データを２秒分読み込む
  %
  wavtmp1l = fread( fw1lid, m, 'short'); wavtmp1r = fread( fw1rid, m, 'short');
  wavtmp21l = fread( fw21lid, m, 'short'); wavtmp21r = fread( fw21rid, m, 'short');
  wavtmp22l = fread( fw22lid, m, 'short'); wavtmp22r = fread( fw22rid, m, 'short');
  wavtmp23l = fread( fw23lid, m, 'short'); wavtmp23r = fread( fw23rid, m, 'short');
  wavtmp24l = fread( fw24lid, m, 'short'); wavtmp24r = fread( fw24rid, m, 'short');
  wavtmp25l = fread( fw25lid, m, 'short'); wavtmp25r = fread( fw25rid, m, 'short');
  wavtmp26l = fread( fw26lid, m, 'short'); wavtmp26r = fread( fw26rid, m, 'short');

  %  各データを足し合わせる
  %
  p_wavel = wavtmp1l + wavtmp21l + wavtmp22l + wavtmp23l + wavtmp24l + wavtmp25l + wavtmp26
  p_waver = wavtmp1r + wavtmp21r + wavtmp22r + wavtmp23r + wavtmp24r + wavtmp25r + wavtmp26

  %  直接音のみのものをファイルに保存
```

```
  %
  fwrite(fdlid, wavtmp1l, 'short');
  fwrite(fdrid, wavtmp1r, 'short');

  %  直接音＋反射音をファイルに保存
  %
  fwrite(fclid, p_wavel, 'short');
  fwrite(fcrid, p_waver, 'short');

end

%  ファイルポインタを閉じる
%
fclose('all');


%%%%% End of file %%%%%
```

外部関数 drpt.m のプログラムソース

```
function [x] = drpt(spo, obs, room)
%
%  部屋の大きさ、音源位置、観測点の位置より
%  直接音と各反射音の到達距離、方位を鏡像法により求める
%
%  source      spo[a b c]
%  observer    obs[x y z]
%  room       room[l m n]



%
%  直接音      仮想音源（実際の音現位置）(a, b, c)
%

%  到達距離を求める
%
dis1 = dist( (spo(1)-obs(1)), (spo(2)-obs(2)), (spo(3)-obs(3)) );

%  仰角を求める
%
e_ang1 = phi( (spo(1)-obs(1)), (spo(2)-obs(2)), (spo(3)-obs(3)) );
e_ang1 = e_ang1 * (180/pi);    %  rad -> deg

%  水平角を求める
%
a_ang1 = theta( (spo(1)-obs(1)), (spo(2)-obs(2)) );
a_ang1 = a_ang1 * (180/pi);    %  rad -> deg



%
%  反射音 1      仮想音源 (-a, b, c)
%

%  到達距離を求める
%
dis21 = dist( (-spo(1)-obs(1)), (spo(2)-obs(2)), (spo(3)-obs(3)) );

%  仰角を求める
%
```

```
e_ang21 = phi( (-spo(1)-obs(1)), (spo(2)-obs(2)), (spo(3)-obs(3)) );
e_ang21 = e_ang21 * (180/pi);   %  rad -> deg

%  水平角を求める
%
a_ang21 = theta( (-spo(1)-obs(1)), (spo(2)-obs(2)) );
a_ang21 = a_ang21 * (180/pi);   %  rad -> deg



%
%  反射音2     仮想音源 (2l-a, b, c)
%

%  到達距離を求める
%
dis22 = dist( (2*room(1)-spo(1)-obs(1)), (spo(2)-obs(2)), (spo(3)-obs(3)) );

%  仰角を求める
%
e_ang22 = phi( (2*room(1)-spo(1)-obs(1)), (spo(2)-obs(2)), (spo(3)-obs(3)) );
e_ang22 = e_ang22 * (180/pi);   %  rad -> deg

%  水平角を求める
%
a_ang22 = theta( (2*room(1)-spo(1)-obs(1)), (spo(2)-obs(2)) );
a_ang22 = a_ang22 * (180/pi);   %  rad -> deg



%
%  反射音3     仮想音源 (a, -b, c)
%

%  到達距離を求める
%
dis23 = dist( (spo(1)-obs(1)), (-spo(2)-obs(2)), (spo(3)-obs(3)) );

%  仰角を求める
%
e_ang23 = phi( (spo(1)-obs(1)), (-spo(2)-obs(2)), (spo(3)-obs(3)) );
e_ang23 = e_ang23 * (180/pi);   %  rad -> deg
```

```
%  水平角を求める
%
a_ang23 = theta( (spo(1)-obs(1)), (-spo(2)-obs(2)) );
a_ang23 = a_ang23 * (180/pi);    %  rad -> deg



%
%  反射音 4      仮想音源 (a, 2m-b, c)
%

%  到達距離を求める
%
dis24 = dist( (spo(1)-obs(1)), (2*room(2)-spo(2)-obs(2)), (spo(3)-obs(3)) );

%  仰角を求める
%
e_ang24 = phi( (spo(1)-obs(1)), (2*room(2)-spo(2)-obs(2)), (spo(3)-obs(3)) );
e_ang24 = e_ang24 * (180/pi);    %  rad -> deg

%  水平角を求める
%
a_ang24 = theta( (spo(1)-obs(1)), (2*room(2)-spo(2)-obs(2)) );
a_ang24 = a_ang24 * (180/pi);    %  rad -> deg



%
%  反射音 5      仮想音源 (a, b, -c)
%

%  到達距離を求める
%
dis25 = dist( (spo(1)-obs(1)), (spo(2)-obs(2)), (-spo(3)-obs(3)) );

%  仰角を求める
%
e_ang25 = phi( (spo(1)-obs(1)), (spo(2)-obs(2)), (-spo(3)-obs(3)) );
e_ang25 = e_ang25 * (180/pi);    %  rad -> deg

%  水平角を求める
%
a_ang25 = theta( (spo(1)-obs(1)), (spo(2)-obs(2)) );
```

```
a_ang25 = a_ang25 * (180/pi);   %  rad -> deg



%
%  反射音 6      仮想音源 (a, b, 2n-c)
%

%  到達距離を求める
%
dis26 = dist( (spo(1)-obs(1)), (spo(2)-obs(2)), (2*room(3)-spo(3)-obs(3)) );

%  仰角を求める
%
e_ang26 = phi( (spo(1)-obs(1)), (spo(2)-obs(2)), (2*room(3)-spo(3)-obs(3)) );
e_ang26 = e_ang26 * (180/pi);   %  rad -> deg

%  水平角を求める
%
a_ang26 = theta( (spo(1)-obs(1)), (spo(2)-obs(2)) );
a_ang26 = a_ang26 * (180/pi);   %  rad -> deg



%
%  得られた各データを行列型にして引き渡す
%
dis_ang1 = [dis1 e_ang1 a_ang1];
dis_ang21 = [dis21 e_ang21 a_ang21];
dis_ang22 = [dis22 e_ang22 a_ang22];
dis_ang23 = [dis23 e_ang23 a_ang23];
dis_ang24 = [dis24 e_ang24 a_ang24];
dis_ang25 = [dis25 e_ang25 a_ang25];
dis_ang26 = [dis26 e_ang26 a_ang26];

x =[ dis_ang1 ;
     dis_ang21;
     dis_ang22;
     dis_ang23;
     dis_ang24;
     dis_ang25;
     dis_ang26];
```

```
%
%  仰角を求めるためのサブルーチン
%
function a = phi(x, y, z)
%
bunbo = sqrt( x^2 + y^2 );
bunsi = z;
a = atan( bunsi / bunbo );


%
%  水平角を求めるためのサブルーチン
%
function b = theta(x, y)
%
if (y > 0)
   b = 2*pi - acos( x / sqrt( x^2 + y^2 ) );
else
   b = acos( x / sqrt( x^2 + y^2 ) );
end

%
%  2点間の距離を求めるためのサブルーチン
%
function [r] = dist(a, b, c)
%
r = sqrt( a^2 + b^2 + c^2);


%%%%% End of file %%%%%
```

外部関数 imp.m のプログラムソース

```
function[fin] = imp(elev, azim)
%
%  データファイルにない HRTF を周波数領域で比例配分して求める
%

%
%  各仰角の分解能
%
elevs = [-40 -30 -20 -10   0  10  20  30  40  50  60  70  80  90;
    6.43 6.00 5.00 5.00 5.00 5.00 5.00 6.00 6.43 8.00 10.00 15.00 30.00 360];


%
%  HRTF の入力パス設定
%
root = '/home/is/yosuke-t/works/kemar';
dir_ch = '/';
ext = '.dat';


%
%  求めたい角度の近傍の角度を求める
%


%  仰角成分 (phia >= phib)
%
x = round(elev/10)*10;
if (mod(elev, 10) < 5)
   phia = x + 10;
   phib = x;
else
   phia = x;
   phib = x - 10;
end
if (mod(elev, 10) == 0)
   phia = x;
   phib = x;
end
if (phia < -40 | phib <-40)
   phia = -40;
```

```
   phib = -40;
end


%  水平角成分 (thea >= theb)
%

a1 = max( (elevs(1,:) == phia).* elevs(2,:) ); % get point of elevs
if ( mod(azim, a1) == 0)
   thea1 = azim;
   theb1 = azim;
else
   for n1 = 0:round(360/a1)
      if (azim < a1*n1), break, end
   end
   thea1 = round(a1 * n1);
   theb1 = round(a1 * (n1-1));
end
if (thea1 < 0)
   thea1 = 360 + thea1;
end
 if (theb1 < 0)
   theb1 = 360 + theb1;
end


a2 = max( (elevs(1,:) == phib).* elevs(2,:) ); % get point of elevs

if ( mod(azim, a2) == 0)
   thea2 = azim;
   theb2 = azim;
else
   for n2 = 1:round(360/a2)
      if (azim < a2*n2), break, end
   end
   thea2 = round(a2 * n2);
   theb2 = round(a2 * (n2-1));
end
if (thea2 < 0)
   thea1 = 360 + thea1;
end
 if (theb2 < 0)
   theb1 = 360 + theb1;
```

```
end


%
%  必要な4組のHRTFをロードする
%
pathname = hpath(root, dir_ch, 'full', 'L', ext, phia, thea1);
imp11l = rraw(pathname)';
pathname = hpath(root, dir_ch, 'full', 'R', ext, phia, thea1);
imp11r = rraw(pathname)';
pathname = hpath(root, dir_ch, 'full', 'L', ext, phia, theb1);
imp12l = rraw(pathname)';
pathname = hpath(root, dir_ch, 'full', 'R', ext, phia, theb1);
imp12r = rraw(pathname)';

pathname = hpath(root, dir_ch, 'full', 'L', ext, phib, thea2);
imp21l = rraw(pathname)';
pathname = hpath(root, dir_ch, 'full', 'R', ext, phib, thea2);
imp21r = rraw(pathname)';
pathname = hpath(root, dir_ch, 'full', 'L', ext, phib, theb2);
imp22l = rraw(pathname)';
pathname = hpath(root, dir_ch, 'full', 'R', ext, phib, theb2);
imp22r = rraw(pathname)';


%
%  水平角成分で合成 (azim part)
%

% imp11 * imp12 -> impe1
prma1 = (azim - thea2) / a1;
impe1l = real( ifft( (1-prma1)*fft(imp11l) + prma1*fft(imp12l) ) );
impe1r = real( ifft( (1-prma1)*fft(imp11r) + prma1*fft(imp12r) ) );

% imp21 * imp22 -> impe2
prma2 = (azim - thea2) / a2;
impe2l = real( ifft( (1-prma2)*fft(imp21l) + prma2*fft(imp22l) ) );
impe2r = real( ifft( (1-prma2)*fft(imp21r) + prma2*fft(imp22r) ) );


%
%  仰角成分で合成 (elev part)
%
```

```
% impe1 * impe2 -> (l/r)imp
prme = (elev - phib) / 10;
leftimp = real( ifft( (1-prme)*fft(impe1l) + prme*fft(impe2l) ) );
rightimp = real( ifft( (1-prme)*fft(impe1r) + prme*fft(impe2r) ) );



%
%  求めたHRTFを行列形式で返す
%
fin = [leftimp ; rightimp];



%%%%% End of file %%%%%
```

外部関数 hpath.m のプログラムソース

```
function [s] = hpath(root,dir_ch,subdir,select,ext,elev,azim)
%
% function [s] = hrtfpath(root,dir_ch,subdir,select,ext,elev,azim)
% Return pathanme for HRTF data file:
% root is root directory.
% dir_ch is directory character, '/' (unix) or ':' (mac).
% subdir is 'compact', 'full', etc.
% select is 'L', 'R' or 'H'.
% ext is the filename extension '.dat', etc.
% elev is elevation.
% azim is azimuth.
%
s = sprintf('%s%s%s%selev%d%s%s%de%03da%s',...
root,dir_ch,subdir,dir_ch,round(elev),...
dir_ch,select,round(elev),round(azim),ext);



%%%%% End of file %%%%%
```

外部関数 dtime.m のプログラムソース

```
function n = dtime(r)
% dtime dtime(r)
%  サンプリング周波数 44.1[KHz] 時における
%  埋め込むゼロ信号数を求める

sv = 340; % sound verosity[m/s]
sr = 44100; % sampling rate [Hz]

n = round(r / sv * sr);


%%%%% End of file %%%%%
```