ATRテクニカルレポート表紙

[公開]

TR-M-0049

マラソンシステムにおける音響効果
－走行速度に応じた速度で後方へ移動する声援－

二宮　知子　Tomoko NINOMIYA
西村　竜一　Ryouichi NISHIMURA

1999.9.24

ATR 知能映像通信研究所

学外実習報告書
マラソンシステムにおける音響効果
—走行速度に応じた速度で後方へ移動する声援—

奈良先端科学技術大学院大学情報科学研究科
二宮 知子

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　1999年8月2日～9月24日

1　はじめに

バーチャルリアリティ（仮想現実感）とは、人間の知覚系に人工の媒体（メディア）を通じて環境情報を伝達し、被験者に対し物体や現象の存在を仮想的に知覚させる技術である。

人間の知覚には視覚・聴覚・触覚・嗅覚・味覚などがあるが、環境情報の大半を視覚から得ている。そのため現在のバーチャルリアリティは視覚情報により仮想環境を構築するものが多い。近年バーチャルリアリティが注目されるようになったのは三次元画像やコンピュータグラフィックスの技術、画像処理能力の飛躍的な向上によりインタラクティブな画像生成が可能となったからである。しかし、現状の技術では、人が満足できるような臨場感は実現できていない。なぜならば、実像と機器に投影された映像とは、明らかに識別できるからである。その他の臨場感に欠ける理由としては、触覚・力覚や物理法則による拘束の欠如、自然界とのインタラクション、聴覚の欠如が挙げられる。

聴覚は視覚に次いで多くの環境情報を得るため、臨場感を強化するためには視覚に加え聴覚の刺激を与えることが効果的であると考えられる。聴覚を導入する利点としては次のようなことがある。視覚はその有効範囲を視線の方向に限られているが、聴覚は全方位を知覚することができる。また、音は「もの」に状態変化、すなわち運動が生じたときに発生する。よって仮想環境において何らかの作業・動作を行ったとき、音が存在するほうが自然である。さらに映像の「もの」の移動と共に音の移動があれば、印象も強く現実感も増す。付け加えて、オーディオ技術の向上により、現在では生の音と録音された音との差がほとんど区別できなくなり、より自然に近い音が再現可能になったことも挙げられる。

一方、視覚以外の感覚系をも含めた仮想環境を実現し、それら感覚系からの刺激を他の人と共有することで、人と人とのコミュニケーションを支援するシステムの構築が行われている。その一実現例として「マラソンシステム」がある。このシステムは、実在のマラソン選手が実際にマラソンを走ったときの感覚と同じ感覚を、ユーザが仮想空間内で追体験することにより、そのマラソン選手がその時に感じていた喜びや辛さといった気持ちを、ユーザにも感じてもらうこと目指したものである。

この「マラソンシステム」においては、走者の息使いや足音といったものから、風きり音・並走している車やバイクの音・沿道の声援まで、様々な聴覚刺激が必要である。実際には実験室内に構築される仮想空間を、臨

場感溢れるものにするためには、これらの中でも特に沿道の声援を付与することが効果的だと考えられる。本実習では、実際には定位置で走行運動をしているユーザに対し、走行速度に応じた速度で前方から後方へ流れている聴覚刺激（沿道の声援）を提供する仕組みを構築することを目的とした。

2　仮想音場構築の原理

2.1　方向知覚と頭部伝達関数

聴覚による方向知覚では、左右の耳の位置の差により音源からの距離が異なるため、それぞれの耳に届く音には音圧差（強度差）と位相差（時間差）が生じ、これが重要な手がかりとなる。しかしこれらだけでは左右の方向感だけの手がかりとなり、前後・上下といった3次元的な方向の識別はできない。これを識別する要因のひとつが、胴体・頭部・耳介などによる音波の回折によるスペクトルの変化であると考えられる。

反射音がなく受聴者と音源が離れている場合、低い周波数においては回折効果が小さいため音圧差は無視できるが、左右の耳に届く音波に位相差が生じ、これが手がかりとなる。また高い周波数においては位相差が360度以上となり識別できなくなるが、頭部や耳介による回折効果によって音圧差が生じ、これが手がかりとなる。このようなことから、人間の耳に到達する音の音圧差や位相差、スペクトルの変化といった情報を含んだ頭部伝達関数（hrtf）を原音に付加する（畳み込む）ことにより、仮想の音源位置を知覚させることができる。

一般に頭部伝達関数は、人間の頭部を模したダミーヘッド（擬似頭）の両耳の鼓膜位置にマイクロホンを取り付け、無響室においてインパルス応答を計測することによって得られる。しかし、頭部や耳介の形状は個人によって異なるため、一般的な形状（ダミーヘッド）に基づいた頭部伝達関数を用いても音像がうまく定位しない場合もある。

今回用いた頭部伝達関数はMITがWWW上で公開しているものである。ダミーヘッドを中心に、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応答を計測しており、サンプリング周波数は44.1KHzで、仰角-40度から90度内の710点で計測されている。特に今回使用した水平面内の分解能は5度である。

2.2　再生方法

仮想の音場を構築する場合、スピーカ再生とヘッドホン再生が考えられる。スピーカ再生の場合、受聴者が受け取る音はおかれている実環境の影響を受ける。また図1のH_{21}、H_{12}のように、左スピーカの音が右耳へ届き、右スピーカの音が左耳へ届くという空間のクロストークもおこる。これらは各スピーカにフィルタを与えることにより除去できる。ヘッ

ドホン再生の場合、スピーカ再生時のような問題は起こらず、ダミーヘッドによって収録した音をそのまま再生すれば、音場が再現できる。しかし、ヘッドホン特性の混入やヘッドホンを頭部に装着するという肉体的な拘束の問題が起こる。

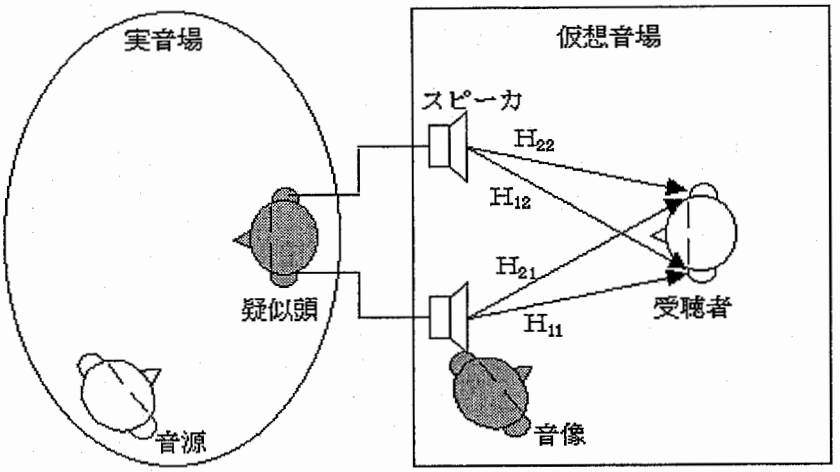

図 1: スピーカ再生

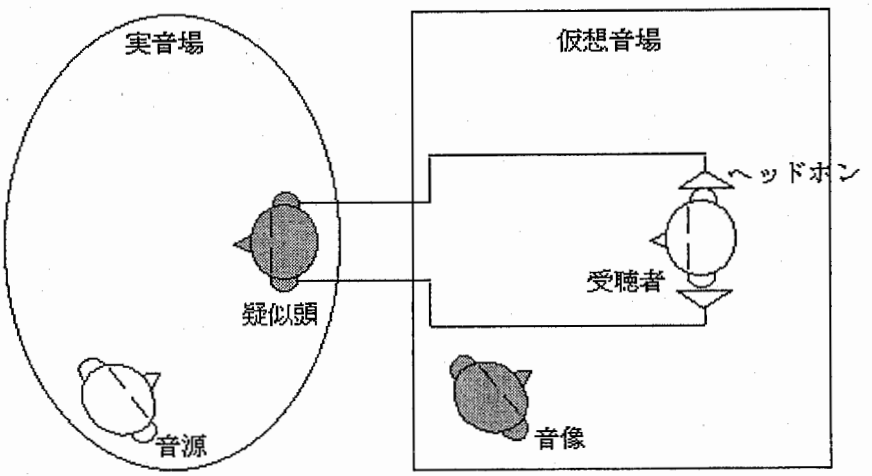

図 2: ヘッドホン再生

3　走行速度に応じた速さで移動する音

3.1　はじめに

ヘッドホン再生による、走者の走行速度に応じた速さで移動する聴覚刺激の実現は、以下のような手順で行う。

- 所望の地点に音像が定位する音（仮想音源）を作成する。
- 隣り合った地点の音を連続して再生する。
- 各地点に対する音の再生時間は走行速度に応じて調整する。

3.2　仮想音源の作成

仮想音源は、原音にその位置に対する伝達特性を畳み込むことによって作成される。しかし、今回用いようとした頭部伝達関数はもともと、ダミーヘッドを中心に半径1.4 mの円周上の音源に対して収録されたものである。そのため、マラソンシステムで必要となる沿道の声援のような、受聴者の横一直線上の音を作るには、この頭部伝達関数に手を加える必要がある。すなわち、距離減衰と空気吸音減衰を付加することである。距離減衰とは、音の強さが音源からの距離の２乗に反比例することを言う。また空気吸音減衰とは媒質による音波の吸収を言い、高い周波数で急速に増大する。

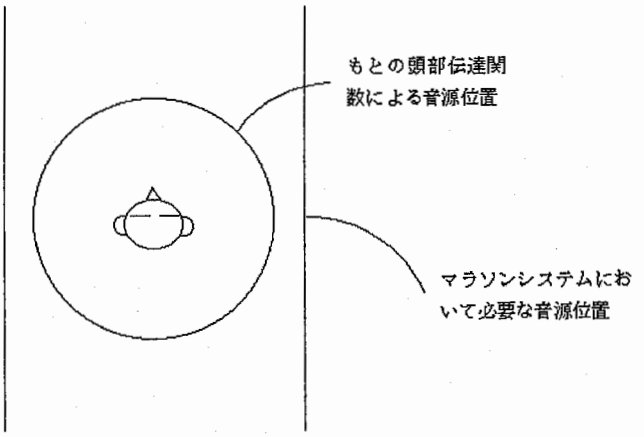

図 3: 所望の音像位置

3.3 新たな頭部伝達関数の作成

まず受聴者を中心とし、1.4mの円周上のある点（角度）と中心を結んだ延長線を引く。その線と受聴者の向きと平行な直線が交わる点を仮想音源の位置（音像を定位させたい位置）とした。この仮想音源の位置は片側19箇所、両側で38箇所を想定した。

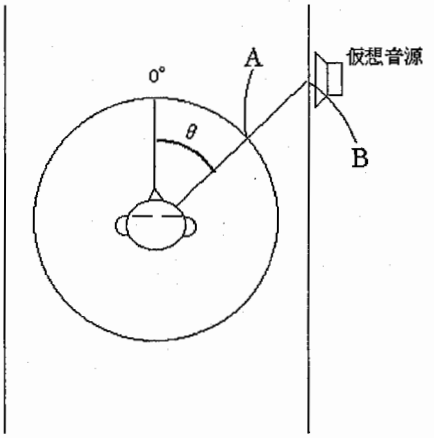

図 4: 仮想音源の位置

仮想音源の位置を円周上の点 (図4の点A) から側方一直線上の点 (図4の点B) へと変更するために、もとの頭部伝達関数に、距離減衰と空気吸音減衰を付加し、距離感を加えた新たな伝達関数を作成する。このとき、1.4 mの円周上のある位置（角度）から左右の耳へ届く頭部伝達関数に減衰を付加しただけでは、それぞれの伝達関数は距離感を持つが、図5のように両者は半径1.4 mの円周上に焦点を結ぶことになる。そのため、1.4 m先にある音の音量が小さくなっただけのように感じ、距離感が変わらない。そこで、図6のように、仮想音源の位置と両耳を結んだ線が半径1.4 mの円と交わる点 (θ_l, θ_r) を求め、θ_l 点から左耳へ、θ_r 点から右耳への頭部伝達関数を減衰させる。すると仮想音源の位置に焦点を結び距離感が生まれる。

3.4 音の移動感

以上のようにして作成した頭部伝達関数を原音に畳み込むことにより、各位置の仮想音源が作成できる。そしてそれらの音を連続して再生する

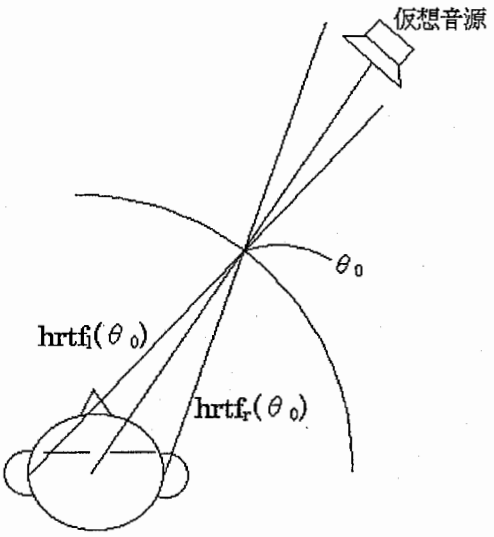

図 5: 距離感を考慮していない伝達関数

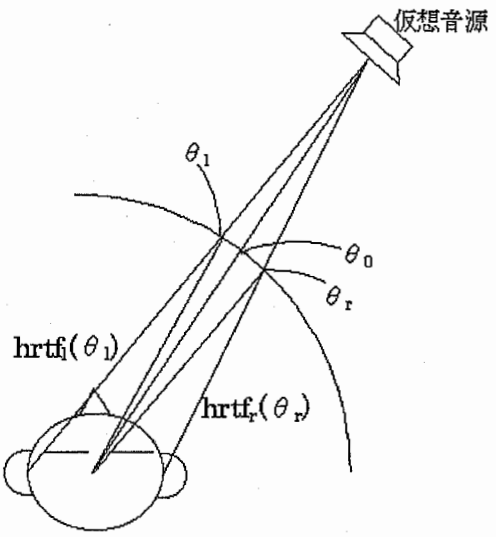

図 6: 距離感を考慮した伝達関数

ことにより移動感を表現する。

図 7 に示すように、各位置に対して一通り用意されている音を少しずつ切り出してきて繋ぐことにより、あたかも一通りの音の中で音源が移動しているかのように感じさせる。

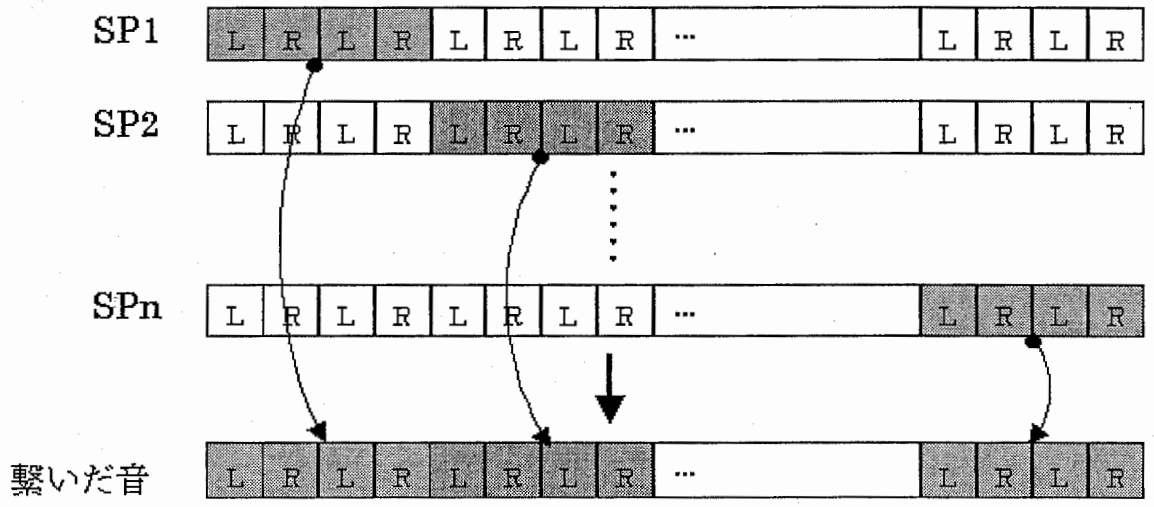

図 7: 音の切り出し

3.5 音のスピード感

音を繋ぐ時、各位置に対する仮想音源の再生時間を調節することで走行速度に応じたスピード感を表現する。まず受聴者の側方に、図8に示すような仮想音源が存在し得る範囲があるとする。今回はその範囲内の19箇所を仮想音源の位置と想定したが、図8では簡単のため4箇所の表示とする。次に各位置にある仮想音源がカバーする範囲を決める。走行速度がわかっていればある時間に進む距離がわかるので、その中にどの仮想音源のカバー範囲がどれだけ含まれているか調べ、各仮想音源の再生時間を計算する。そして上で述べたように音を繋ぐ。

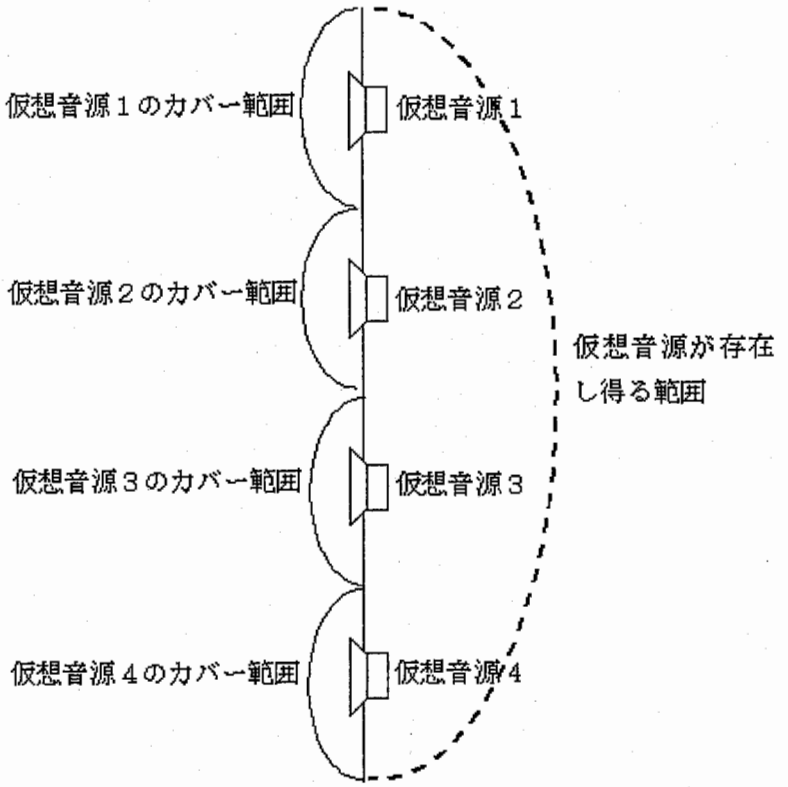

図 8: スピーカの担当範囲

4　スピーカ再生の検討

ここまで述べてきた作業を施した音を、そのままヘッドホンにより再生すれば走行速度に応じて音が移動するような仮想環境は構築できる。しかし、「マラソンシステム」における受聴者は走行しているため、ヘッドホンを装着するといった、身体的な制約は少ないほうがよいと考えられる。そこで、スピーカによる仮想環境の構築を検討した。

スピーカにより仮想環境を構築する際には、スピーカから制御点（今回は受聴者の両耳）までの伝達特性や、空間のクロストークを考慮しなければならない。この問題をクリアするには2つのアプローチが考えられる。ひとつは、空間の伝達特性やクロストークをすべてキャンセルするようなフィルタを作成しスピーカの前段に入れる方法である。もうひとつは伝達特性やクロストークが加わることによって、目的とする伝達特性を模擬するようなフィルタを作成しスピーカの前段に入れる方法である。今回は後者の方法を用いた。

4.1　フィルタの生成

図9に4chスピーカによる伝達特性の模擬系を示す。ここでX_jは制御用フィルタの伝達関数、H_{ij}制御用スピーカから両耳までの伝達関数、D_iは模擬する音源から両耳までの伝達関数、Sは入力信号のスペクトルを表している。

この伝達系を式を用いて表すと式(1)のようになる。

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \end{bmatrix} \begin{bmatrix} X_1S \\ X_2S \\ X_3S \\ X_4S \end{bmatrix} = \begin{bmatrix} D_1S \\ D_2S \end{bmatrix} \tag{1}$$

ここでSは両辺に共通なので、Sで両辺を割ると次式となる。

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \tag{2}$$

それぞれのスピーカの前段に入れる制御用フィルタ X_j は式 (2) を解くことによって求められる。

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{21} \\ H_{12} & H_{22} \\ H_{13} & H_{23} \\ H_{14} & H_{24} \end{bmatrix} \left\{ \begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \end{bmatrix} \begin{bmatrix} H_{11} & H_{21} \\ H_{12} & H_{22} \\ H_{13} & H_{23} \\ H_{14} & H_{24} \end{bmatrix} \right\}^{-1} \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \quad (3)$$

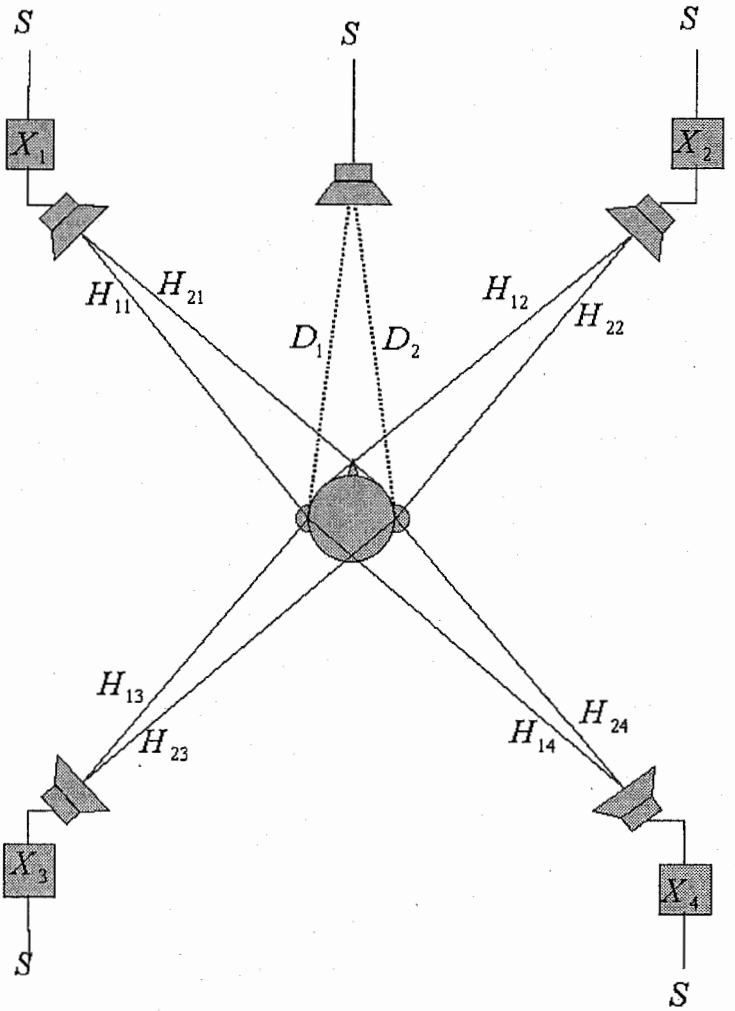

図 9: 4ch スピーカによる伝達特性の模擬系

4.2 検討

得られた X_j を用いて 4ch スピーカにより再現実験を行ったが、再現精度が低かった。この原因として考えられることは 2 点ある。ひとつは各仮想音源位置からの伝達関数を作成する際に室内伝達特性を考慮しなかったことである。このため再生環境の伝達特性の影響を受け、厳密な再現ができなかったと考えられる。もうひとつは受聴者の両耳鼓膜近傍の 2 点のみを制御点としたことである。受聴者が頭部を動かすと、受聴者の耳は制御点からはずれ、その耳に届く音は意図したものとまったく異なる音となり、再現ができなくなったと考えられる。

この結果から、今回はシステムの完成が主眼であるため、スピーカ再生にはこだわらずヘッドホン再生によりシステムを構築することとした。

5　処理の流れ

以下に今回作成したヘッドホン再生によるシステムの処理の流れを示す。

受聴者の左側から聞こえてくる音と、右側から聞こえてくる音に対する走行速度は別々に取得できるようにしてみた。これは、カーブに対応できるひとつの手段となるかもしれないと考えたからである。

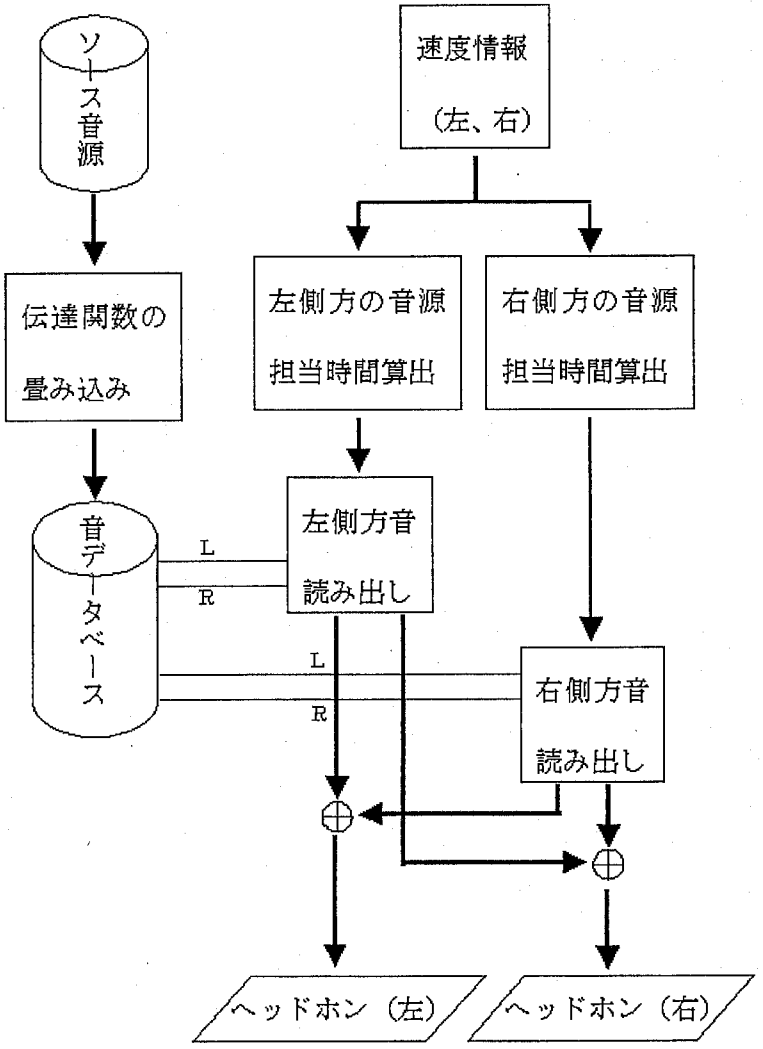

図 10: 処理の流れ

6　まとめ

今回、厳密な聴取実験は行っていないが、数人に試聴してもらい簡単に感想を述べてもらった。その結果ほとんどの人は音源が移動して感じられるという感想であった。よって本実習の目的である、走行速度に応じて移動する聴覚刺激を提供する仕組みは構築できたと思われる。

しかし、屋外で聞こえるような音には感じられず、臨場感に欠けるという意見もあった。これは原音を屋外で収録した音にするなど原音自体の改良が考えられる他、走者自身の足音を加えたり、遠くにあって移動しない定常音を加えるなどの工夫が必要である。

また、走行速度が速くなりすぎると音が不自然につながって聞こえるという問題も発生した。これは走行速度が速くなると、各位置の仮想音源の再生時間が極端に短くなるため発生すると考えられる。そこで、スピードに応じて各位置の仮想音源がカバーする範囲を調整し、再生時間調整することで回避できるのではないだろうか。

受聴者の身体的制約を軽減するためのスピーカ再生については、頭部の位置ずれに強い模擬系を構成する方法（AMLSE）などを取り入れていくことが必要だと考えられる。

付録（プログラム）

受聴者より1.4m離れた側方一直線上から到来する頭部伝達関数を作成する（matlab）

```
function hrtfnew2(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 被験者から1.4m離れて、一直線に前から後ろへ流れるHRTFを作る
% s:角度（HRTFの角度指定と同様）
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% すべての角度、空気吸音（温度10℃ 相対湿度50％）+HRTFの補間
%

%
% HRTFと同様の角度指定を、三角関数と同様の角度指定にする
if (0<=s)&(s<90),
    ss=90-s;
  elseif (90<=s)&(s<180),
    ss=s-90;
  elseif (180<=s)&(s<=270),
    ss=270-s;
  elseif (270<s)&(s<360),
    ss=s-270;
end;

%
% 受聴者と新たな仮想音源との距離
m=1.4/cos(2*pi*ss/360);

%
% 新たな仮想音源の位置と受聴者の両耳を結んだ時、半径1.4mの円周上
% のどの角度になるか、また、新たな仮想音源と各耳との距離を求める
[l,r,zl,zr,xl,xr]=kakudo(m,s);
```

```
%
% 求めた点のHRTFを補間して作る
hl=hokanL1(l);
hr=hokanR1(r);

%
% 空気減衰関数の読み込み（絶対値になおしたもの）
fid=fopen('/home/is/tomoko-n/atr/matlab/dec2','r');
g=fread(fid,'double');
fclose(fid);



bl = 1-((1-g)*(zl-xl)/100);
br = 1-((1-g)*(zr-xr)/100);

%
% 時間領域に変換
cl=ifft(bl);
cr=ifft(br);

%
% 実数部をとる
dl=real(cl);
dr=real(cr);

%
% 補間したHRTFと空気吸音減衰関数の畳み込み
sl=conv(hl,dl);
sr=conv(hr,dr);

%
% 距離の2乗に反比例
sl=sl/((zl/xl)^2);
sr=sr/((zr/xr)^2);
```

```
%
% 出来上がった新たな頭部伝達関数を書き込む
bufl=sprintf('/home/is/tomoko-n/hrtfn-11/elev0/L0e%03.0fa.dat',s);
bufr=sprintf('/home/is/tomoko-n/hrtfn-11/elev0/R0e%03.0fa.dat',s);

fid=fopen(bufl,'w');
fwrite(fid,sl,'double');
fclose(fid);

fid=fopen(bufr,'w');
fwrite(fid,sr,'double');
fclose(fid);
```

側方一直線上から左耳へ到来するHRTFが、1.4mの円周上のどの点を通るか算出する（matlab）

```
function [sl1,z2]=kakudoL(b,do)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%左耳に届く音がr1.4mの円上でどの角度にあるか求める
% この時の角度[do]三角関数を考える時と同様
% b:離したい距離 do:角度
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% 元の距離
%
a=1.4;

%
% 頭の幅の半分
%
c=0.1;

%
% 聞こえさせたい角度
%
s=2*pi*do/360;

%
% 角度と距離を求める。
%
x2=b*cos(s);
y2=b*sin(s);
z2=sqrt(y2^2 + c^2 + 2*c*x2 + x2^2);

s4=asin(y2/z2);
x3=c*cos(s4);
y3=c*sin(s4);
s5=2*pi*90/360-s4;
```

```
s7=acos(y3/a);

sl=2*pi*180/360-(s5+s7);

sl1=sl*360/(2*pi);
```

側方一直線上から右耳へ到来するHRTFが、1.4mの円周上のどの点を通るか算出する（matlab）

```
function [sr1,z1]=kakudoR(b,do)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%右耳に届く音がR1.4mの円上でどの角度にあるか求める
% この時の角度[do]は三角関数を考える時と同様
% b:離したい距離 do:角度
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
%元の距離%
%
a=1.4;

%
%頭の幅の半分
%
c=0.1;

%
%聞こえさせたい角度
%
s=2*pi*do/360;

%
% 角度と距離を求める。
%
x1=c*cos(s);
y1=c*sin(s);
z1=sqrt(y1^2+(b-x1)^2);

s3=acos(y1/z1);

k=[1 -2*c*sin(s-s3) c^2-a^2];
r=roots(k);
```

```
%
% ここでrのうち正の数を選ぶ ＝k1
%
r1=r(1,1);
r2=r(2,1);

if r1>0,k1=r1;
elseif r2>0,k1=r2;
else, break,end

i=k1*sin(s-s3);
h=k1*cos(s-s3);
sr=asin(h/a);
sr1=sr*360/(2*pi);
```

kakudoLにより求めた点から左耳へ到来するHRTFを2点線形補間により算出（matlab）

```
function z2 = hokanL1(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HRTFの補間をする
% s:角度（実数）（HRTFの角度指定と同様）
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% kakudo.mで出した角度を3桁の整数に直す。
%
a = sprintf('%03.0f', s);
b = sscanf(a, '%d');

q = rem(b, 5);

%
% 補間したい角度より小さいHRTFの角度を求める
i = b-q;
%
% 補間したい角度より大きいHRTFの角度を求める
j=i+5;

%
% 整数にしたら5度きざみと差がなかったとき。そのまま終了
%
if q == 0
  v= sprintf('/home/is/tomoko-n/atr/myhrtf/elev0/L0e%03.0fa.dat',b);
  fid = fopen(v, 'r');
  z2 = fread(fid, 'double');
  fclose(fid);
  break;

else
  %
```

```
  % もし大きい方の角度が３６０度になったら０度に直す
  if (j==360)
    j=0;
  end

  %
  % 補間したい角度に近い大小の５度きざみのhrtfを読んでくる
  %
  m = sprintf('/home/is/tomoko-n/atr/myhrtf/elev0/L0e%03.0fa.dat',i);
  n = sprintf('/home/is/tomoko-n/atr/myhrtf/elev0/L0e%03.0fa.dat',j);

  fid=fopen(m,'r');
  x1=fread(fid,'double');
  fclose(fid);

  fid=fopen(n,'r');
  x2=fread(fid,'double');
  fclose(fid);

  %
  % 読んできたHRTFを周波数領域に変換
  %
  y1=fft(x1);
  y2=fft(x2);

  %
  % 上で出しておいた比率に従って２点線形補間
  %
  z1= (5-q)/5*y1 + q/5*y2;

  %
  % 時間領域に変換
  %
  z2=real(ifft(z1));
end
```

kakudoRにより求めた点から右耳へ到来するHRTFを2点線形補間により算出（matlab）

```
function z2 = hokanR1(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%HRTFの補間をする
% s:角度（実数）（HRTFの角度指定と同様）
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% kakudo.mで出した角度を3桁の整数に直す。
%
a = sprintf('%03.0f', s);
b = sscanf(a, '%d');

q = rem(b, 5);

%
% 補間したい角度より小さいHRTFの角度を求める
i = b-q;
%
% 補間したい角度より大きいHRTFの角度を求める
j=i+5;

%
% 整数にしたら5度きざみと差がなかったとき。そのまま終了
%
if q == 0
  if b == 0
    b = 360
  end
  v= sprintf('/home/is/tomoko-n/atr/myhrtf/elev0/L0e%03.0fa.dat',360-b);
  fid = fopen(v, 'r');
  z2 = fread(fid, 'double');
  fclose(fid);
  break;
```

```
else
  %
  % 補間したい角度に近い大小の5度きざみのhrtfを読んでくる
  %
  if i == 0
    i = 360;
  end
  if j == 0
    j = 360
  end

  m = sprintf('/home/is/tomoko-n/atr/myhrtf/elev0/L0e%03.0fa.dat',360-i);
  n = sprintf('/home/is/tomoko-n/atr/myhrtf/elev0/L0e%03.0fa.dat',360-j);

  fid=fopen(m,'r');
  x1=fread(fid,'double');
  fclose(fid);

  fid=fopen(n,'r');
  x2=fread(fid,'double');
  fclose(fid);

  %
  % 読んできたHRTFを周波数領域に変換
  %
  y1=fft(x1);
  y2=fft(x2);

  %
  % 上で出しておいた比率に従って2点線形補間
  %
  z1= (5-q)/5*y1 + q/5*y2;

  %
```

```
  % 時間領域に変換
  %
  z2=real(ifft(z1));

end
```

L,R別々になっているデータをひとつのステレオデータにまとめる（matlab）

```
function zz=stereo(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2つのスピーカの音を一つのファイルに直す %
% i：角度指定 （HRTFの角度指定方法）      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% L、R各ファイル名の読み込み
%
l = sprintf('speakerL/sounds/gunsyuM/L0e%03.0fa.raw',s); % mono left sound
r = sprintf('speakerR/sounds/gunsyuM/R0e%03.0fa.raw',s); % mono right sound

%
% ステレオ音の書き込み先ファイル名の指定
%
s = sprintf('STEREO/sounds/gunsyuM/S0e%03.0fa.raw',s); % stereo sound

%
% L音ファイルを開いてデータを読み込む
%
fid = fopen(l,'r');
a = fread(fid,'short');     % open and read L sound
fclose(fid);

%
% R音ファイルを開いてデータを読み込む
%
fid = fopen(r,'r');
b = fread(fid,'short');     % open and read R sound
fclose(fid);

%
% モノラル音のサイズ取得
```

```
%
k = size(a,1);

%
% ステレオ音のサイズ分の領域確保
%
z = zeros(2,k);   % make 2-by-k matrix of zeros

%
% L音とR音で2-by-kのマトリックスを作る
%
z(1,:) = reshape(a,1,k);
z(2,:) = reshape(b,1,k);  % make 2-by-k matrix from L and R

%
% 2-by-k のマトリックスを 1-by-2*k に変形
%
zz = reshape(z,k*2,1);  % make 2*K-by-1 matrix

%
% ステレオ音の書き込み
%
fid = fopen(s,'w');
fwrite(fid,zz,'short');   % open and write stereo sound
fclose(fid);
```

4chスピーカ再生用のフィルタを作成する（matlab）

```
function haibun2(s)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4ch スピーカ用のフィルタを作る
% pinvを使って最小ノルム解で求める
% スピーカの位置は全部45度方向とする%
% sp1->L:H11
% sp1->R:H12
% sp2->L:H21
% sp2->R:H22
% sp3->L:H31
% sp3->R:H32
% sp4->L:H41
% sp4->R:H42
%
% s:角度指定（HRTFの角度指定と同様）
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% 現実のスピーカ１から左耳への伝達関数のファイルを開く
%
fid=fopen('hrtfn/elev0/L0e315a.dat','r');
h11=fread(fid,'double');
fclose(fid);

%
% 現実のスピーカ１から右耳への伝達関数のファイルを開く
%
fid=fopen('hrtfn/elev0/R0e315a.dat','r');
h12=fread(fid,'double');
fclose(fid);

%
% 現実のスピーカ２から左耳への伝達関数のファイルを開く
%
```

```
fid=fopen('hrtfn/elev0/L0e045a.dat','r');
h21=fread(fid,'double');
fclose(fid);

%
% 現実のスピーカ2から右耳への伝達関数のファイルを開く
%
fid=fopen('hrtfn/elev0/R0e045a.dat','r');
h22=fread(fid,'double');
fclose(fid);

%
% 現実のスピーカ3から左耳への伝達関数のファイルを開く
%
fid=fopen('hrtfn/elev0/L0e225a.dat','r');
h31=fread(fid,'double');
fclose(fid);

%
% 現実のスピーカ3から右耳への伝達関数のファイルを開く
%
fid=fopen('hrtfn/elev0/R0e225a.dat','r');
h32=fread(fid,'double');
fclose(fid);

%
% 現実のスピーカ4から左耳への伝達関数のファイルを開く
%
fid=fopen('hrtfn/elev0/L0e135a.dat','r');
h41=fread(fid,'double');
fclose(fid);

%
% 現実のスピーカ4から右耳への伝達関数のファイルを開く
%
```

```
fid=fopen('hrtfn/elev0/R0e135a.dat','r');
h42=fread(fid,'double');
fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% 角度 s 地点からの HRTF の読み込み
%

%
% 左耳への HRTF
l=sprintf('hrtfn/elev0/L0e%03.0fa.dat',s);
fid=fopen(l,'r');
hl=fread(fid,'double');
fclose(fid);

%
% 右耳への HRTF
r=sprintf('hrtfn/elev0/R0e%03.0fa.dat',s);
fid=fopen(r,'r');
hr=fread(fid,'double');
fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% 読み込んだ各データを周波数領域に変換%
%
H11=fft(h11);
H12=fft(h12);

H21=fft(h21);
H22=fft(h22);
```

```
H31=fft(h31);
H32=fft(h32);

H41=fft(h41);
H42=fft(h42);


HL=fft(hl);
HR=fft(hr);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% H=現実のスピーカからの伝達関数、T=仮想音源位置からのHRTF、S=求
めたいフィルタ
%

for i=1:575
  H=[H11(i),H21(i),H31(i),H41(i);
     H12(i),H22(i),H32(i),H42(i)];


  T=[HL(i);
     HR(i)];

  S=pinv(H)*T;

  S1(i)=S(1);
  S2(i)=S(2);
  S3(i)=S(3);
  S4(i)=S(4);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
% 求めたフィルタを時間領域に戻す %
%
s1=real(ifft(S1));
s2=real(ifft(S2));
s3=real(ifft(S3));
s4=real(ifft(S4));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% 求めたフィルタを書き込む
%

%
% 現実のスピーカ1の前段に入れるフィルタ
a=sprintf('speaker1/filter2/elev0/S0e%03.0fa.dat',s);
fid=fopen(a,'w');
fwrite(fid,s1,'double');
fclose(fid);

%
% 現実のスピーカ2の前段に入れるフィルタ
b=sprintf('speaker2/filter2/elev0/S0e%03.0fa.dat',s);
fid=fopen(b,'w');
fwrite(fid,s2,'double');
fclose(fid);

%
% 現実のスピーカ3の前段に入れるフィルタ
c=sprintf('speaker3/filter2/elev0/S0e%03.0fa.dat',s);
fid=fopen(c,'w');
fwrite(fid,s3,'double');
fclose(fid);
```

```
%
% 現実のスピーカ４の前段に入れるフィルタ
d=sprintf('speaker4/filter2/elev0/S0e%03.0fa.dat',s);
fid=fopen(d,'w');
fwrite(fid,s4,'double');
fclose(fid);
```

作成した音をスピード調節しながら連続して再生する。左右の音の鳴り始めるタイミングはランダム（c）

```
/* 走行速度に応じた速さで移動する音源 */
/* 左右の音が出るタイミングはランダム */
/* 標準入力を見に行って、左右の速度を取得する */
/* 音源 gunsyuM */

#include <stdio.h>          /* printf 等でいる*/
#include <dmedia/audio.h> /* オーディオライブラリ等 */
#include <sys/types.h>     /**/
#include <sys/stat.h>      /* ここの３つは open 等で使う */
#include <fcntl.h>         /**/
#include <unistd.h>        /* seek 等で使う */
#include <math.h>          /* 三角関数など */
#include <stdlib.h>        /* 乱数 */
#include <sys/time.h>      /* gettimeofday に使う */


#define IsLtrgOn (RandLOn | disposingLOn)/* 左側の音を出すトリ
ガ */
#define IsRtrgOn (RandROn | disposingROn)/* 右側の音を出すトリ
ガ */

#define pi 3.1416
#define Dislarge(x) 1.4*(tan(2*pi*(x+5)/360)-tan(2*pi*x/360))/*
担当範囲を決め*/
                                                              /*
るための計算 */
#define Dissmall(x) 1.4*(tan(2*pi*x/360)-tan(2*pi*(x-5)/360))

#define sampling 44100
#define SIZE 1593472 /* 音のデータのサイズ。バイト数で */


main()
```

```
{
  int err;

/* オーディオポートの準備 */
  ALconfig audioconfig;
  ALport audioport;

/* トリガの一つ（乱数）の初期化 */
  char RandLOn = 0x00;
  char RandROn = 0x00;

/* トリガの一つ（作動中）の初期化 */
  char disposingLOn = 0x00;
  char disposingROn = 0x00;

/* 各地点の音の情報の構造体 */
  typedef struct {
    int flg;
    char SpName[256];
    double SpAxisfrom;
    double SpAxisto;
    double Spdis;
    int fd;
  }t_dfile;

/* 左右の音の構造体 */
  t_dfile ftblL[20];
  t_dfile ftblR[20];

/* ファイル開く用 */
  int index=0;
  int s,ss;
  double m1,m2;
```

```
  struct timeval timev[2];  /* 現在時刻の取得用 */
                            /* time[0]=前の時刻 ,time[1]=今の
時刻 */
  double passtime;          /* passtime=経過時間  */
  double tu1,tu0;           /* timev によって得られた時刻 (ms)
部分を */
                            /* 1000 分の 1 する */
  double time1,time0;       /* timev によって得られた時刻を通常
の */
                            /* 現在時刻にして取得 */
  double ls[20];            /* 再生時の各仮想音源の担当距離 */
  double timeS[20];         /* 再生時の各仮想音源の担当時間 */
  double l;                 /* 1 ループの間に進む距離 */
  double rem=0;             /* 1ループの間に進む距離から各スピー
カの */
                            /* 担当範囲を計算した時のあまり */
  double Llnow,Rlnow;       /* 左右の音の現在位置 */

  double vl,vr;             /* 左右の速度 */
  double LMAX;              /* 音が聞こえる範囲の長さ */
  int i,j,k,h,p,a;
  int SpNo,SpStart;
  int nbyte,nokori,tugi;

  short bufL[sampling*2*3],bufR[sampling*2*3],bufS[sampling*2*3];
                            /* 音を入れる配列 */

  int offsetL = 0,offsetR = 0;  /* offset 用 */
  int head;                 /* 音をつなぐ時の配列の先頭 */
  int hd = 0;

  double X,Y;
  long tl;                  /* 乱数の種用 */
  long filled;
```

```
  double spdl,spdr;
  fd_set readfds;
  struct timeval timeout={0,0};

  /****************** ファイルを全部開く ********************************/

  printf("file open start\n");

  ftblR[0].SpAxisfrom = 0;  /* 最初のスピーカの担当区間の開始点 */
  ftblL[0].SpAxisfrom = 0;

  for(s=45;s<90;s+=5){      /* 右側前方 */
    ss = 90-s;

    ftblR[index].flg=1;     /* スイッチ */

    sprintf(ftblR[index].SpName,"/home/is/tomoko-n/atr/jikken/white/elev0/S0e%(
                            /* ファイル名 */
    m1 = Dislarge(ss)/2;    /* 自分の前方のスピーカとの距離の半
分 */
    m2 = Dissmall(ss)/2;    /* 自分の後方のスピーカとの距離の半
分 */
    ftblR[index].Spdis = m1+m2;  /* 担当距離 */

    ftblR[index].SpAxisto = ftblR[index].SpAxisfrom + ftblR[index].Spdis;
                            /*このスピーカの担当終了点 */
    ftblR[index+1].SpAxisfrom = ftblR[index].SpAxisto;
                            /* 次スピーカの担当開始点 */

    ftblR[index].fd = open(ftblR[index].SpName,O_RDONLY);  /* フ
ァイルを開く */

    index++;                /* スピーカ番号をインクリメント */
  }
```

```
  if(s==90){                    /* 右側真横 */
    ss = 0;                     /* 以下、同様 */
    ftblR[index].flg = 1;
    sprintf(ftblR[index].SpName,"/home/is/tomoko-n/atr/jikken/white/elev0/S0e
    ftblR[index].Spdis = Dislarge(ss);
    ftblR[index].SpAxisto = ftblR[index].SpAxisfrom + ftblR[index].Spdis;
    ftblR[index+1].SpAxisfrom = ftblR[index].SpAxisto;
    ftblR[index].fd = open(ftblR[index].SpName,O_RDONLY);
    index++;
  }


  for(s=95;s<=135;s+=5){        /* 右側後方 */
    ss = s-90;                  /* 以下同様 */
    ftblR[index].flg=1;
    sprintf(ftblR[index].SpName,"/home/is/tomoko-n/atr/jikken/white/elev0/S0e
    m1 = Dislarge(ss)/2;
    m2 = Dissmall(ss)/2;
    ftblR[index].Spdis = m1+m2;
    ftblR[index].SpAxisto = ftblR[index].SpAxisfrom + ftblR[index].Spdis;
    ftblR[index+1].SpAxisfrom = ftblR[index].SpAxisto;
    ftblR[index].fd = open(ftblR[index].SpName,O_RDONLY);

    LMAX = ftblR[index].SpAxisto;  /* 距離の最大値 */
    index++;
  }
  ftblR[index].Spdis = ftblR[index-1].Spdis;  /* 最後方のみせ
かけのスピーカ */
                                              /* 終りかどうか
のチェック用 */
  index = 0;
  for(s=315;s>270;s-=5){        /* 左側前方 */
    ss = s-270;                 /* 以下同様 */
    ftblL[index].flg=1;
    sprintf(ftblL[index].SpName,"/home/is/tomoko-n/atr/jikken/white/elev0/S0e
```

```
    m1 = Dislarge(ss)/2;
    m2 = Dissmall(ss)/2;
    ftblL[index].Spdis = m1+m2;
    ftblL[index].SpAxisto = ftblL[index].SpAxisfrom + ftblL[index].Spdis;
    ftblL[index+1].SpAxisfrom = ftblL[index].SpAxisto;
    ftblL[index].fd = open(ftblL[index].SpName,O_RDONLY);
    index++;
  }

  if(s==270){                    /* 左側真横 */
    ss = 0;                      /* 以下同様 */
    ftblL[index].flg = 1;
    sprintf(ftblL[index].SpName,"/home/is/tomoko-n/atr/jikken/white/elev0/S0e%(
    ftblL[index].Spdis = Dislarge(ss);
    ftblL[index].SpAxisto = ftblL[index].SpAxisfrom + ftblL[index].Spdis;
    ftblL[index+1].SpAxisfrom = ftblL[index].SpAxisto;
    ftblL[index].fd = open(ftblL[index].SpName,O_RDONLY);
    index++;
  }

  for(s=265;s>=225;s-=5){        /* 左側後方 */
    ss = 270-s;                  /* 以下同様 */
    ftblL[index].flg=1;
    sprintf(ftblL[index].SpName,"/home/is/tomoko-n/atr/jikken/white/elev0/S0e%(
    m1 = Dislarge(ss)/2;
    m2 = Dissmall(ss)/2;
    ftblL[index].Spdis = m1+m2;
    ftblL[index].SpAxisto = ftblL[index].SpAxisfrom + ftblL[index].Spdis;
    ftblL[index+1].SpAxisfrom = ftblL[index].SpAxisto;
    ftblL[index].fd = open(ftblL[index].SpName,O_RDONLY);
    index++;
  }

  ftblL[index].Spdis = ftblL[index-1].Spdis;  /* 最後方のみせ
かけのスピーカ */
```

```
                                                    /* 終りかどうか
のチェック用 */
  printf("file open finish\n");

/*------------------------ ファイル開き終り ---------------------------------*/


/* 出力用オーディオポートの準備 */
  audioconfig = ALnewconfig();
  audioport = ALopenport("test2","w",audioconfig);

/* オーディオポートのエラー検出 */
  if(audioport==(ALport)0){
    err=oserror();
    if(err==AL_BAD_NO_PORTS)
      printf("!1\n");
    else if(err==AL_BAD_DEVICE_ACCESS)
      printf("!2\n");
    else if(err==AL_BAD_OUT_OF_MEM)
      printf("!3\n");
  }

/* 左右のスピードの取得（初回） */
  printf("Get speed value (l r)= ");
  fscanf(stdin,"%lf %lf",&spdl,&spdr);
  vl=spdl/10;
  vr=spdr/10;

  printf("vl=%lf\n",vl);
  printf("vr=%lf\n",vr);

/* 現在時刻の取得 */
  gettimeofday(&timev[0]);
```

```
/***************************** ループ開始 *********************************/


  while(1){
    /* 標準入力から入力があったか調べるための初期化 */
    FD_ZERO(&readfds);
    FD_SET(0,&readfds);

    /* 標準入力から入力があったか調べる */
    if(select(1,&readfds,NULL,NULL,&timeout)){
      fscanf(stdin,"%lf %lf",&spdl,&spdr);

      /* 入力された値が 0<=(vl,vr)<=10 だったら作動。それ以外は
終了 */
      if((0<=spdl && spdl<=10) && (0<=spdr && spdr<=10)){
vl=spdl/10;
vr=spdr/10;

printf("vl=%lf\n",vl);
printf("vr=%lf\n",vr);
      }
      else{
printf("Bud speed value\n");
break;
      }
    }

    /* メモリ領域確保 */
    memset(bufL,0,sizeof(bufL));
    memset(bufR,0,sizeof(bufR));
    memset(bufS,0,sizeof(bufS));

    /* 乱数の種 */
    tl=time(NULL);
    srand48(tl);
```

```
/* 乱数の取得X（左用） */
X=drand48();

if( X<0.2){
  RandL0n = 0x01;
}
else
  RandL0n = 0x00;

/* 乱数の取得Y（右用） */
Y=drand48();

if(Y<0.2){
  RandR0n = 0x01;
}
else
  RandR0n = 0x00;

/* 現在時刻の取得 */
gettimeofday(&timev[1]);

/* 現在時刻の１０００分の１秒の部分を取得 */
tu0 = (double)timev[0].tv_usec/1000000.;
tu1 = (double)timev[1].tv_usec/1000000.;

/* 現在時刻をマイクロセカンドで取得 */
time0 = (double)timev[0].tv_sec + tu0;
time1 = (double)timev[1].tv_sec + tu1;

/* 経過時間の算出 */
passtime =time1-time0;

/* 現在の時刻を過去の時刻とする */
timev[0] = timev[1];
```

```
/*++++++++++++++++++++++ 左側の音を読む ++++++++++++++++++++++++++++++*/

    if(IsLtrgOn){
      head = 0;

      /* 現在位置の初期化 */
      if(disposingLOn == 0x00)
        Llnow = 0;

      /* 移動距離の計算 */
      l=vl*passtime;

      /* 全体の距離を超えたら終了サイン */
      if(Llnow+l >= LMAX){
        l = LMAX-Llnow;
        disposingLOn = 0x00;
      }
      /* 配置中サイン */
      else
disposingLOn = 0x01;

      /* lnowはどのスピーカの範囲か？ */
      for(i=0;ftblL[i].SpAxisto < Llnow;i++)
;  /* 何もせず、直下へ */

      SpNo = i;
      SpStart = i;

      /* 移動距離 l が、あるスピーカの範囲内の時 */
      if((Llnow+l) < ftblL[SpNo].SpAxisto){
lseek(ftblL[SpStart].fd,offsetL,SEEK_SET);
nbyte = (int)(sampling * passtime) * 2 * 2;
offsetL += nbyte;
```

```
/* 音源のサイズ（時間）を超えるとき */
if(offsetL > SIZE){
  nokori = SIZE-(offsetL-nbyte);
  tugi = offsetL - SIZE;

  hd = head/2;
  read(ftblL[SpNo].fd,&bufL[hd],nokori);
  head += nokori;

  offsetL = 0;
  lseek(ftblL[SpNo].fd,offsetL,SEEK_SET);
  read(ftblL[SpNo].fd,&bufL[hd],tugi);
  head += tugi;
  offsetL += tugi;
}
/* 音源のサイズ（時間）を超えないとき */
else
  read(ftblL[SpNo].fd,bufL,nbyte);
Llnow += 1;
      }

      /* 二個以上のスピーカの範囲にまたがる場合 */
      else{
while((rem =(Llnow+1)-ftblL[SpNo].SpAxisto) > ftblL[SpNo+1].Spdis){
  /* 移動範囲内で0番目の仮想音源の担当距離 */
  if(j == 0){
    ls[j] = ftblL[SpNo].SpAxisto - Llnow;
    j++;
    SpNo++;
  }
  /* 1番目以降の距離 */
  else{
    ls[j] = ftblL[SpNo].Spdis;
    j++;
```

```
    SpNo++;
  }
}

/* 各スピーカが担当する距離からそれぞれの時間を計算 */
for(k=0;k<j;k++)
  timeS[k] = (ls[k]/l)*passtime;

timeS[k] = (rem/l)*passtime;

/* 時間分の音を読み込みつなぐ */
for(h=0;h<=k;h++){
  lseek(ftblL[SpStart].fd,offsetL,SEEK_SET);
  nbyte = (int)(sampling * timeS[h])*2*2;
  offsetL += nbyte;

  /* 音源のサイズ（時間）を超えるとき */
  if(offsetL > SIZE){
        nokori = SIZE-(offsetL-nbyte);
    tugi = offsetL - SIZE;

    hd = head/2;
    read(ftblL[SpStart].fd,&bufL[hd],nokori);
    head += nokori;

    offsetL = 0;
    lseek(ftblL[SpStart].fd,offsetL,SEEK_SET);
    hd = head/2;
    read(ftblL[SpStart].fd,&bufL[hd],tugi);
    head += tugi;
    offsetL += tugi;
  }
  /* 音源のサイズ（時間）を超えないとき */
  else{
    hd = head/2;
```

```
    read(ftblL[SpStart].fd,&bufL[hd],nbyte);
    head += nbyte;
  }

  SpStart++;
}
      }
      Llnow += l;
    }  /* if(IsLtrgOn) 文の終了 */


/*+++++++++++++++++ 右側の音を読む ++++++++++++++++++++++++++++++++++++++++++++++++*/

    if(IsRtrgOn){
      head = 0;

      if(disposingROn == 0x00)
Rlnow = 0;

      /* 距離の計算 */
      l=vr*passtime;

      /* 全体の距離を超えたら終了サイン */
      if(Rlnow+l >= LMAX){
l = LMAX-Rlnow;
disposingROn = 0x00;
      }
      /* 配置中サイン */
      else
disposingROn = 0x01;

      /* lnowはどのスピーカの範囲か？ */
      for(i=0;ftblR[i].SpAxisto < Rlnow;i++)
;  /* 何もせず直下へ */
```

```
        SpNo = i;
        SpStart = i;

        /* 移動距離 l が、あるスピーカの範囲内の時 */
        if((Rlnow+l) < ftblR[SpNo].SpAxisto){
lseek(ftblR[SpStart].fd,offsetR,SEEK_SET);
nbyte = (int)(sampling * passtime) * 2 * 2;
offsetR += nbyte;

/* 音源のサイズ（時間）を超えるとき */
if(offsetR > SIZE){
  nokori = SIZE-(offsetR-nbyte);
  tugi = offsetR - SIZE;

  hd = head/2;
  read(ftblR[SpNo].fd,&bufR[hd],nokori);
  head += nokori;

  offsetR = 0;
  lseek(ftblR[SpNo].fd,offsetR,SEEK_SET);
  read(ftblR[SpNo].fd,&bufR[hd],tugi);
  head += tugi;
  offsetR += tugi;
}
/* 音源のサイズ（時間）を超えないとき */
else
  read(ftblR[SpNo].fd,bufR,nbyte);
Rlnow += l;
        }

        /* 二個以上のスピーカの範囲にまたがる場合 */
        else{
while((rem = (Rlnow+l)-ftblR[SpNo].SpAxisto) > ftblR[SpNo+1].Spdis){
  /* 0番目の距離 */
  if(j == 0){
```

```
    ls[j] = ftblR[SpNo].SpAxisto - Rlnow;
    j++;
    SpNo++;
  }
  /* 1番目以降の距離 */
  else{
    ls[j] = ftblR[SpNo].Spdis;
    j++;
    SpNo++;
  }
}

/* 各スピーカが担当する距離からそれぞれの時間を計算 */
for(k=0;k<j;k++)
  timeS[k] = (ls[k]/l)*passtime;

timeS[k] = (rem/l)*passtime;

 /* 時間分の音を読み込みつなぐ */
for(h=0;h<=k;h++){
  lseek(ftblR[SpStart].fd,offsetR,SEEK_SET);
  nbyte = (int)(sampling * timeS[h])*2*2;
  offsetR += nbyte;

  /* 音源のサイズ（時間）を超えるとき */
  if(offsetR > SIZE){
    nokori = SIZE-(offsetR-nbyte);
    tugi = offsetR - SIZE;

    hd = head/2;
    read(ftblR[SpStart].fd,&bufR[hd],nokori);
    head += nokori;

    offsetR = 0;
    lseek(ftblR[SpStart].fd,offsetR,SEEK_SET);
```

```
    hd = head/2;
    read(ftblR[SpStart].fd,&bufR[hd],tugi);
    head += tugi;
    offsetR += tugi;
  }
  /* 音源のサイズ（時間）を超えないとき */
  else{
    hd = head/2;
    read(ftblR[SpStart].fd,&bufR[hd],nbyte);
    head += nbyte;
  }

  SpStart++;
}
      }
      Rlnow += 1;
    }  /* if(IsRtrgOn)文の終了 */


/*+++++++++++++++ LとRのファイルを足す ++++++++++++++++++++++++++++++++++++++++*/

    for(p=0;p<(int)(sampling*passtime)*2;p++)
      bufS[p]=bufL[p]+bufR[p];

    /* オーディオポートが開かなかったとき */
    if(audioport == 0){
      printf("audioport can't open\n");
      ALcloseport(audioport);
      ALfreeconfig(audioconfig);
      break;
    }
    /* 音の出力 */
    else
      ALwritesamps(audioport,bufS,(int)(sampling*passtime)*2);
```

```
  }
/******************** while(1)の終了 ***************************************/

  /* 音のファイルを閉じる */
  for(a=0;a<20;a++)
    close(ftblL[a].fd);
  for(a=0;a<20;a++)
    close(ftblR[a].fd);

  /* オーディオポートを閉じる */
  do
    filled = ALgetfilled(audioport);
  while(filled > 0);
  ALcloseport(audioport);
  ALfreeconfig(audioconfig);

}  /* main の終了 */
```