LAMPIRAN

1. Form Bimbingan Skripsi
2. Source Code : interp3d_noplot.m
3. Source Code : make_hrtf_interp_rect.m
4. Source Code : make_hrtf_interp_tri.m
5. Source Code : make_mse_hrir_mph_rect.m
6. Source Code : make_sd_interprect_hrtf.m

Source Code : interp3d_noplot.m

```
function interp3d_noplot

    %INITIALIZE
    % Flags
    global USE_ADJACENCY_WALK
        USE_ADJACENCY_WALK= true;  % true:  use adjacency walk
                    % false: use brute-force search

        USE_OCTREE= true;         % true:  use Octree query to find starting
                    %       tetrahedron for adjacency walk
                    % false: use random starting tetrahedron for
                    %       adjacency walk

    % constants
    binCapacity = 5;          % maximum number of points stored in each leaf
                    % node of the octree
    global fs
        fs = 65536;               % sampling rate [Hz]

    %%%% INITIALISATIONS %%%%

    % get HRTF measurement coordinates
    %disp('Retrieving HRTF measurement coordinates...');
    aziEleDist0 = get_HRTF_coords;

    global T;
    global X;
    global N;
    global OT;
    % Generate tetrahedral mesh via Delaunay triangulation
    %disp('Generating tetrahedral mesh and adjacency map of measurement
points...');
    if str2double(datestr(datenum(version('-date')),'YYYY'))>=2013
       dt = delaunayTriangulation(aziEleDist0);
       T = dt.ConnectivityList;    % contains the tetrahedral mesh
       X = dt.Points;              % contains the vertices of the tetrahedra
       N = neighbors(dt);          % contains the adjacency list
    else
       % for older Matlab releases < 2013
       T = delaunay(aziEleDist0(:,1), aziEleDist0(:,2), aziEleDist0(:,3));
       X = aziEleDist0;
       N = dtNeighbours(T);
    end

    if USE_OCTREE
```

```
        if ~USE_ADJACENCY_WALK
            warning('Octree search can only be used in conjunction with adjacency
walk, otherwise the selection might not terminate...');
            USE_ADJACENCY_WALK = true;
        end

        % Generate octree
        %disp('Generating octree...');

        OT = getOT(aziEleDist0, binCapacity);

        % get centres and one tetrahedron per octree leaf node
        OT = getOTcentres(OT, T);

        % find close tetrahedron
        %disp('Querying octree...');

    else
        ti = 1;
    end

    %%Interpolating the HRTF/HRIR begins here
    load coordinates_interp3d.mat;
    for i = 1:length(aziEleDist3d)
      koord = squeeze(aziEleDist3d(i,:));
      main_loop(koord(1),koord(2),koord(3));
      fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',koord(3),koord(2),koord(1),koord(2),koord(3));
    end

end

function main_loop(a,e,d)
    global USE_ADJACENCY_WALK;
    global T;
    global X;
    global N;
    global OT;

    % variable initialisations
    source_pos = [a,e,d]; % desired source position (azimuth [deg], elevation
[deg], radius [cm])
    ti = queryOT(source_pos, OT);
    disp(['Using tetrahedron ', num2str(ti), ' as starting point for adjacency
walk']);
    Niter = 0;
```

```
tetra_indices = zeros(size(T,1),1);
MIN_GAIN = -0.00001;        % due to numerical issues, very small
negative
                    % gains are considered 0


%%%% MAIN LOOP %%%%

% iterate through mesh to find tetrahedron for interpolation
%disp(['Iterating through ', num2str(size(T,1)), ' tetrahedra...']);
for t = 1:size(T,1)
    % count iterations
    Niter = Niter+1;

    % vertices of tetrahedron
    HM = X(T(ti,:),:);
    v4 = HM(4,:);
    H = HM(1:3,:) - repmat(v4,3,1);
    tetra_indices(t) = ti;

    % calculate barycentric coordinates
    bary_gains = [(source_pos-v4)*(H^-1), 0];
    bary_gains(4) = 1-sum(bary_gains);

    % check barycentric coordinates
    if all(bary_gains>=MIN_GAIN)
        bary_gains = max(bary_gains, 0);
        disp('Success!');
        %disp(['Tetrahedron ID: ', num2str(ti), ', iterations: ', num2str(Niter)]);
        tetra_indices = tetra_indices(1:Niter);
        break;
    end

    if USE_ADJACENCY_WALK
        % move to adjacent tetrahedron
        [tmp, bi] = min(bary_gains);
        ti = N(ti,bi);
    else
        % BRUTE-FORCE: move to next tetrahedron in list
        ti = ti+1;
    end
end

if tetra_indices(end)==0
    error('No tetrahedron found. Exiting...');
end
```

```
    % ambil data HRIR
    HM = X(T(tetra_indices(end),:),:);
    hrir_r = zeros(1024,4);
    hrir_l = zeros(1024,4);
    HRTF_L = zeros(1024,4);
    HRTF_R = zeros(1024,4);
    for hi = 1:4
        azi = HM(hi,1);
        ele = HM(hi,2);
        dist = HM(hi,3);
        %r = HM(hi,3)/100;
        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele,azi,ele,dist);
        eval(load_hrir);
        %aziInteraural = asind(sind(azi)*cosd(ele));   % convert to interaural
coordinates
        %distance_attenuation = 1/r;
        hrir_l(:,hi) = hrir_mph_l;
        HRTF_L(:,hi) = fft(hrir_l(:,hi));
        hrir_r(:,hi) = hrir_mph_r;
        NFFT = size(HRTF_R,1);
        HRTF_R(:,hi) = fft(hrir_r(:,hi));
    end

    %%%% INTERPOLATE %%%%
    %disp('>>> Interpolation: <<<');
    %disp('bary_gains * abs(HRTF)');
    hrir_r = transpose(hrir_r);
    hrir_l = transpose(hrir_l);
    hrir_r_interp = bary_gains*hrir_r;
    hrir_l_interp = bary_gains*hrir_l;

    hint_l = bary_gains*abs(HRTF_L(:,:)');
    hint_r = bary_gains*abs(HRTF_R(:,:)');
    %%%%%%%%%%%%%%%%%%%%%
    save_hrtf = sprintf('save
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_interp3d.mat hint_l
hint_r',d,e,a,e,d);
    eval(save_hrtf);
    save_hrir = sprintf('save
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interp3d.mat hrir_l_interp
hrir_r_interp',d,e,a,e,d);
    eval(save_hrir);
    %load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',d,e,a,e,d);
```

```
    %eval(load_hrtf);
    %figure
    %plot(hint_l,'r');
    %hold on;
    %plot(hint_r,'r');
    %hold on;
    %plot(hrtf_mph_l,'b');
    %hold on;
    %plot(hrtf_mph_r,'b');
end




%% Get HRTF coordinates
function aziEleDist0 = get_HRTF_coords
    % Get the HRTF coordinates used in the PKU&IOA HRTF database [1]
    %
    % [1] Qu, T., Xiao, Z., Gong, M., Huang, Y., Li, X., and Wu, X. (2009).
    % "Distance-dependent head-related transfer functions measured with high
    % spatial resolution using a spark gap", IEEE Audio, Speech, Language
    % Process. 17, 1124?1132.

    % Elevations measured
    Elevations = -40:10:90;

    % Azimuth steps from lowest to highest elevation
    AzimuthStep = [repmat(5, 1, 10), 10, 15, 30, 360];

    % Distances measured
    Distances = [20:10:50 , 100, 130, 160];
    %titik 75 diskip karena itu merupakan titik yang dicari

    % Total number of HRTFs in database
    NumHRTFs = sum(360./AzimuthStep)*length(Distances);

    % preallocate output array
    aziEleDist0 = zeros(NumHRTFs,3);

    % MAIN loop
    n = 0;
    for dist = Distances
        elei = 0;
        for ele = Elevations
```

```
            elei = elei+1;
            for azi = 0:AzimuthStep(elei):359
               n = n+1;
               aziEleDist0(n,:) = [azi, ele, dist];
            end
         end
      end

end

%% get OCTREE
% Organise 3-D points pts into an octree data structure
%
% binCapacity: maximum number of points per leaf node
%
% References:
% Samet, H. (1989). "Implementing ray tracing with octrees and neighbor
% finding", Computers & Graphics 13, pp. 445?460.
function OT = getOT(pts, binCapacity)
   % inits
   OT = struct;

   % start with root
   OT.Points = pts;
   OT.PointBins = ones(size(pts,1),1);
   OT.BinBoundaries = [min(pts), max(pts)];
   OT.BinParents = 0;
   OT.BinDepths = 0;

   % MAIN LOOP
   OT = getChildren(OT,1,binCapacity);
   OT.BinCount = size(OT.BinParents,1);

end

% Recursive call to split bin into octants
function OT = getChildren(OT, parent, binCapacity)
   parentBoundaries = OT.BinBoundaries(parent,:);
   for bini = 1:8
      binBoundaries = zeros(1, 6);
      rng = range(reshape(parentBoundaries,3,2),2)';
      binBoundaries(1:3) = bitget(bini-1,1:3).*rng/2 + parentBoundaries(1:3);
      binBoundaries(4:6) = binBoundaries(1:3) + rng/2;

      % add bin to list of bins
      OT.BinBoundaries = [OT.BinBoundaries; binBoundaries];
```

```
        OT.BinParents = [OT.BinParents; parent];
        OT.BinDepths = [OT.BinDepths; OT.BinDepths(parent)+1];
        currBin = size(OT.BinParents,1);

        % check if bin contains any points
        ptinds = (1:size(OT.Points,1))';
        ptinds = ptinds(OT.PointBins==parent);

        binPoints = inBin(OT.Points(ptinds,:), binBoundaries);
        binPoints = ptinds(binPoints);
        if ~isempty(binPoints)
            OT.PointBins(binPoints) = currBin;
            if length(binPoints)>binCapacity
                % keep dividing
                OT = getChildren(OT, currBin, binCapacity);
            end
        end
    end
end

% Determine which points pts are contained in bin
function inds = inBin(pts, binBoundaries)
    N = size(pts,1);
    lpts = pts-repmat(binBoundaries(1:3),N,1);
    upts = pts-repmat(binBoundaries(4:6),N,1);
    inds = find(sum(lpts>=0 & upts<=0,2)==3);
end

% get OT bin centres
function OT = getOTcentres(OT, T)
    NPoints = size(OT.Points,1);
    NBins = OT.BinCount;
    OT.BinCentres = zeros(NBins,3);
    OT.BinChildren = zeros(NBins,8);
    for bi = 1:NBins
        bin_children = find(OT.BinParents==bi);
        if isempty(bin_children)
            % leaf node: insert a random point as a "child"
            pt = find(OT.PointBins==bi,1);
            if isempty(pt)
                % bin contains no point -> find closest point to bin centre
                binb = OT.BinBoundaries(bi,:);
                binc = (binb(1:3)+binb(4:6))./2;
                binn = sqrt(sum( (OT.Points-repmat(binc, NPoints,1)).^2,2 ));
                [tmp, pt] = min(binn);
            end
```

```
            % find a tetrahedron containing pt
            Tind = find(T==pt,1);
            Tind = bmod(Tind, size(T,1));

            OT.BinChildren(bi,2) = Tind;
            continue;
        end
        boundaries = OT.BinBoundaries(bin_children,:);

        % find centre
        x = unique(boundaries(:,[1,4]));
        y = unique(boundaries(:,[2,5]));
        z = unique(boundaries(:,[3,6]));
        OT.BinCentres(bi,:) = [x(2),y(2),z(2)];

        % store children in order
        for bch = 1:8
            chb = OT.BinBoundaries(bin_children(bch),:);
            chc = (chb(1:3)+chb(4:6))./2;
            chind = 0;
            for j = 1:3
                if chc(j) >= OT.BinCentres(bi,j)
                    chind = chind + 2^(j-1);
                end
            end
            OT.BinChildren(bi,chind+1) = bin_children(bch);
        end
    end
end

% query octree
function ti = queryOT(pos, OT)
    binind = 1;
    maxbiniter = 50;
    biniter = 0;
    while (1)
        if (biniter > maxbiniter)
            ti = 1;
            fprintf('No bin found...\n');
            break;
        end

        biniter = biniter + 1;
        currbin = binind;
        octind = 0;
```

```
        binc = OT.BinCentres(binind,:);

        if (pos(1)>=binc(1)); octind = bitor(octind,1); end
        if (pos(2)>=binc(2)); octind = bitor(octind,2); end
        if (pos(3)>=binc(3)); octind = bitor(octind,4); end

        has_children = OT.BinChildren(binind,1)>0;

        if (~has_children)
            ti = OT.BinChildren(currbin,2);
            break;
        end

        binind = OT.BinChildren(binind,octind+1);
    end
end

%% Get adjacency list for delaunay triangulation
% T: connectivity list (tetrahedral mesh) of delaunay triangulation
function N = dtNeighbours(T)

    % create matrix containing all edges/faces
    NT = size(T,1);
    Tdim = size(T,2);
    face = zeros(Tdim*NT,Tdim-1);
    vinds = zeros(Tdim*NT,1);
    Tinds = zeros(Tdim*NT,1);
    ptr1 = 1;
    ptr2 = NT;

    for fi = 1:Tdim
        inds = 1:Tdim;
        inds(fi) = [];
        face(ptr1:ptr2,:) = T(:,inds);
        face(ptr1:ptr2,:) = sort(face(ptr1:ptr2,:), 2);
        vinds(ptr1:ptr2) = fi;
        Tinds(ptr1:ptr2) = 1:NT;
        ptr1 = ptr1+NT;
        ptr2 = ptr2+NT;
    end
    [faceS, indS] = sortrows(face);
    vindsS = vinds(indS);
    TindsS = Tinds(indS);

    N = -ones(size(T));
```

```
    n = 0;
    while n<size(faceS,1)-1
        n = n+1;
        f1 = faceS(n,:);
        f2 = faceS(n+1,:);
        if sum(f1==f2)==(Tdim-1)
            % found matching edge/face!
            t1 = TindsS(n);
            t2 = TindsS(n+1);
            vi1 = vindsS(n);
            vi2 = vindsS(n+1);
            N(t1,vi1) = t2;
            N(t2,vi2) = t1;
            n = n+1;
        end
    end

end

%% alternative modulus function
% mod(N,N) returns N (instead of 0)
% useful for indexing in Matlab
function y = bmod(x,a)
    y = mod(x-1,a)+1;
end
```

Source Code : make_hrtf_interp_rect.m

```
%interprect : using azi and ele
%interprect1: using dist and ele
%interprect2: using dist and azi

load coordinates_interp2d.mat

%azimuth and elevation
for dist = Distances
    elei = 1;
    for ele = Elevations2
        elei = elei+1;
        for azi = AzimuthStep(elei):AzimuthStep(elei):359
            azi0 = azi - 5;
            azi1 = azi + 5;
            ele0 = Elevations(elei-1);
            ele1 = Elevations(elei+1);
            cazi = (azi-azi0) / (azi1-azi0);
            cele = (ele-ele0) / (ele1-ele0);
```

```
        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele0,azi0,ele0,dist);
        fail = false;
        try
            eval(load_hrir);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist,ele0,azi0,ele0,dist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        a_l = (1-cazi)*(1-cele)*hrir_mph_l;
        a_r = (1-cazi)*(1-cele)*hrir_mph_r;

        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele0,azi1,ele0,dist);
        try
            eval(load_hrir);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist,ele0,azi1,ele0,dist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        b_l = (cazi)*(1-cele)*hrir_mph_l;
        b_r = (cazi)*(1-cele)*hrir_mph_r;

        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele1,azi1,ele1,dist);
        try
            eval(load_hrir);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist,ele1,azi1,ele1,dist);
            fail = true;
        end
        if fail == true
```

```
            fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        c_l = (cazi)*(cele)*hrir_mph_l;
        c_r = (cazi)*(cele)*hrir_mph_r;

        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele1,azi0,ele1,dist);
        try
            eval(load_hrir);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist,ele1,azi0,ele1,dist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        d_l = (1-cazi)*(cele)*hrir_mph_l;
        d_r = (1-cazi)*(cele)*hrir_mph_r;

        interp_hrir_l = a_l + b_l + c_l + d_l;
        interp_hrir_r = a_r + b_r + c_r + d_r;
        if fail == false
            save_hrir = sprintf('save
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect.mat interp_hrir_l
interp_hrir_r',dist,ele,azi,ele,dist);
            eval(save_hrir);
        end
        fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
      end
   end
end

%distance and elevation
for dist = Distances
   elei = 1;
   for ele = Elevations2
      elei = elei+1;
      for azi = AzimuthStep(elei):AzimuthStep(elei):359
         ele0 = Elevations(elei-1);
         ele1 = Elevations(elei+1);
```

```
        dist0 = Distances2(1);
        dist1 = Distances2(3);
        cdist = (dist-dist0) / (dist1-dist0);
        cele = (ele-ele0) / (ele1-ele0);
        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist0,ele0,azi,ele0,dist0);
        fail = false;
        try
           eval(load_hrir);
        catch
           fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist0,ele0,azi,ele0,dist0);
           fail = true;
        end
        if fail == true
           fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
           continue;
        end
        a_l = (1-cdist)*(1-cele)*hrir_mph_l;
        a_r = (1-cdist)*(1-cele)*hrir_mph_r;

        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist1,ele0,azi,ele0,dist1);
        try
           eval(load_hrir);
        catch
           fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist1,ele0,azi,ele0,dist1);
           fail = true;
        end
        if fail == true
           fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
           continue;
        end
        b_l = (cdist)*(1-cele)*hrir_mph_l;
        b_r = (cdist)*(1-cele)*hrir_mph_r;

        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist1,ele1,azi,ele1,dist1);
        try
           eval(load_hrir);
        catch
           fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist1,ele1,azi,ele1,dist1);
```

```
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        c_l = (cdist)*(cele)*hrir_mph_l;
        c_r = (cdist)*(cele)*hrir_mph_r;

        load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist0,ele1,azi,ele1,dist0);
        try
            eval(load_hrir);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist0,ele1,azi,ele1,dist0);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        d_l = (1-cdist)*(cele)*hrir_mph_l;
        d_r = (1-cdist)*(cele)*hrir_mph_r;

        interp_hrir_l = a_l + b_l + c_l + d_l;
        interp_hrir_r = a_r + b_r + c_r + d_r;
        if fail == false
            save_hrir = sprintf('save
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect1.mat interp_hrir_l
interp_hrir_r',dist,ele,azi,ele,dist);
            eval(save_hrir);
        end
        fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
      end
   end
end

%distance and azimuth
for dist = Distances
   elei = 1;
   for ele = Elevations2
      elei = elei+1;
```

```
        for azi = AzimuthStep(elei):AzimuthStep(elei):359
            azi0 = azi - 5;
            azi1 = azi + 5;
            dist0 = Distances2(1);
            dist1 = Distances2(3);
            cdist = (dist-dist0) / (dist1-dist0);
            cazi = (azi-azi0) / (azi1-azi0);
            load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist0,ele,azi0,ele,dist0);
            fail = false;
            try
                eval(load_hrir);
            catch
                fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist0,ele,azi0,ele,dist0);
                fail = true;
            end
            if fail == true
                fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
                continue;
            end
            a_l = (1-cdist)*(1-cazi)*hrir_mph_l;
            a_r = (1-cdist)*(1-cazi)*hrir_mph_r;

            load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist1,ele,azi0,ele,dist1);
            try
                eval(load_hrir);
            catch
                fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist1,ele,azi0,ele,dist1);
                fail = true;
            end
            if fail == true
                fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
                continue;
            end
            b_l = (cdist)*(1-cazi)*hrir_mph_l;
            b_r = (cdist)*(1-cazi)*hrir_mph_r;

            load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist1,ele,azi1,ele,dist1);
            try
                eval(load_hrir);
```

```
            catch
               fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist1,ele,azi1,ele,dist1);
               fail = true;
            end
            if fail == true
               fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
               continue;
            end
            c_l = (cdist)*(cazi)*hrir_mph_l;
            c_r = (cdist)*(cazi)*hrir_mph_r;

            load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist0,ele,azi1,ele,dist0);
            try
               eval(load_hrir);
            catch
               fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist0,ele,azi1,ele,dist0);
               fail = true;
            end
            if fail == true
               fprintf('dist%d elev%d hrir_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
               continue;
            end
            d_l = (1-cdist)*(cazi)*hrir_mph_l;
            d_r = (1-cdist)*(cazi)*hrir_mph_r;

            interp_hrir_l = a_l + b_l + c_l + d_l;
            interp_hrir_r = a_r + b_r + c_r + d_r;
            if fail == false
               save_hrir = sprintf('save
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect2.mat interp_hrir_l
interp_hrir_r',dist,ele,azi,ele,dist);
               eval(save_hrir);
            end
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
         end
      end
   end
```

Source Code : make_hrtf_interp_tri.m

```
%interptri : using azi and ele
```

```
%interptri1: using dist and ele
%interptri2: using dist and azi

load coordinates_interp2d.mat

%azimuth and elevation
for dist = Distances
   elei = 1;
   for ele = Elevations2
      elei = elei+1;
      for azi = AzimuthStep(elei):AzimuthStep(elei):359
         azi0 = azi - 5;
         azi1 = azi + 5;
         ele0 = Elevations(elei-1);
         ele1 = Elevations(elei+1);
         dekat1 = abs(azi -azi1);
         dekat2 = abs(azi -azi0);
         if dekat1<dekat2
            cazi = dekat1 / (azi1-azi0);
            dekatAzi = azi1;
            jauhAzi = azi0;
         else
            cazi = dekat2 / (azi1-azi0);
            dekatAzi = azi0;
            jauhAzi = azi1;
         end
         dekat1 = abs(ele -ele1);
         dekat2 = abs(ele -ele0);
         if dekat1<dekat2
            cele = dekat1 / (ele1-ele0);
            dekatEle = ele1;
            jauhEle = ele0;
         else
            cele = dekat2 / (ele1-ele0);
            dekatEle = ele0;
            jauhEle = ele1;
         end
         alp = 1 - cazi - cele;
         %alp -> dekatele dekatazi
         %cele -> jauhele dekatazi
         %cazi -> dekatele jauhazi
         load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dist,dekatEle,dekatAz
i,dekatEle,dist);
         fail = false;
         try
```

```
            eval(load_hrtf);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist,dekatEle,dekatAzi,d
ekatEle,dist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        a_l = alp*hrtf_mph_l;
        a_r = alp*hrtf_mph_r;

        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dist,jauhEle,dekatAzi,
jauhEle,dist);
        try
            eval(load_hrtf);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist,jauhEle,dekatAzi,jau
hEle,dist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        b_l = cele*hrtf_mph_l;
        b_r = cele*hrtf_mph_r;

        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dist,dekatEle,jauhAzi,
dekatEle,dist);
        try
            eval(load_hrtf);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dist,dekatEle,jauhAzi,de
katEle,dist);
            fail = true;
        end
        if fail == true
```

```
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
          end
          c_l = (cazi)*hrtf_mph_l;
          c_r = (cazi)*hrtf_mph_r;


          interp_hrtf_l = a_l + b_l + c_l;
          interp_hrtf_r = a_r + b_r + c_r;
          if fail == false
            save_hrtf = sprintf('save
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_interptri.mat interp_hrtf_l
interp_hrtf_r',dist,ele,azi,ele,dist);
            eval(save_hrtf);
          end
          fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
       end
   end
end

%distance and elevation
for dist = Distances
   elei = 1;
   for ele = Elevations2
      elei = elei+1;
      for azi = AzimuthStep(elei):AzimuthStep(elei):359
         ele0 = Elevations(elei-1);
         ele1 = Elevations(elei+1);
         dist0 = Distances2(1);
         dist1 = Distances2(3);
         dekat1 = abs(dist -dist1);
         dekat2 = abs(dist -dist0);
         if dekat1<dekat2
           cdist = dekat1 / (dist1-dist0);
           dekatDist = dist1;
           jauhDist = dist0;
         else
           cdist = dekat2 / (dist1-dist0);
           dekatDist = dist0;
           jauhDist = dist1;
         end
         dekat1 = abs(ele -ele1);
         dekat2 = abs(ele -ele0);
         if dekat1<dekat2
```

```
            cele = dekat1 / (ele1-ele0);
            dekatEle = ele1;
            jauhEle = ele0;
        else
            cele = dekat2 / (ele1-ele0);
            dekatEle = ele0;
            jauhEle = ele1;
        end
        alp = 1 - cazi - cele;
        %alp -> dekatele dekatdist
        %cele -> jauhele dekatdist
        %cdist -> dekatele jauhdist
        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dekatDist,dekatEle,azi
,dekatEle,dekatDist);
        fail = false;
        try
            eval(load_hrtf);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dekatDist,dekatEle,azi,d
ekatEle,dekatDist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        a_l = alp*hrtf_mph_l;
        a_r = alp*hrtf_mph_r;

        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dekatDist,jauhEle,azi,
jauhEle,dekatDist);
        try
            eval(load_hrtf);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dekatDist,jauhEle,azi,jau
hEle,dekatDist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
```

```
            continue;
        end
        b_l = cele*hrtf_mph_l;
        b_r = cele*hrtf_mph_r;

        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',jauhDist,dekatEle,azi,
dekatEle,jauhDist);
        try
            eval(load_hrtf);
        catch
            fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',jauhDist,dekatEle,azi,de
katEle,jauhDist);
            fail = true;
        end
        if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
        end
        c_l = (cazi)*hrtf_mph_l;
        c_r = (cazi)*hrtf_mph_r;


        interp_hrtf_l = a_l + b_l + c_l;
        interp_hrtf_r = a_r + b_r + c_r;
        if fail == false
            save_hrtf = sprintf('save
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_interptri1.mat interp_hrtf_l
interp_hrtf_r',dist,ele,azi,ele,dist);
            eval(save_hrtf);
        end
        fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
    end
  end
end

%distance and azimuth
for dist = Distances
  elei = 1;
  for ele = Elevations2
    elei = elei+1;
    for azi = AzimuthStep(elei):AzimuthStep(elei):359
      azi0 = azi - 5;
```

```
        azi1 = azi + 5;
        dist0 = Distances2(1);
        dist1 = Distances2(3);
        dekat1 = abs(dist -dist1);
        dekat2 = abs(dist -dist0);
        if dekat1<dekat2
          cdist = dekat1 / (dist1-dist0);
          dekatDist = dist1;
          jauhDist = dist0;
        else
          cdist = dekat2 / (dist1-dist0);
          dekatDist = dist0;
          jauhDist = dist1;
        end
        dekat1 = abs(azi -azi1);
        dekat2 = abs(azi -azi0);
        if dekat1<dekat2
          cazi = dekat1 / (azi1-azi0);
          dekatAzi = azi1;
          jauhAzi = azi0;
        else
          cazi = dekat2 / (azi1-azi0);
          dekatAzi = azi0;
          jauhAzi = azi1;
        end
        alp = 1 - cazi - cele;
        %alp -> dekatazi dekatdist
        %cazi -> jauhazi dekatdist
        %cdist -> dekatazi jauhdist
        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dekatDist,ele,dekatAz
i,ele,dekatDist);
        fail = false;
        try
           eval(load_hrtf);
        catch
           fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dekatDist,ele,dekatAzi,el
e,dekatDist);
           fail = true;
        end
        if fail == true
           fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
           continue;
        end
```

```
        a_l = alp*hrtf_mph_l;
        a_r = alp*hrtf_mph_r;

        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dekatDist,ele,jauhAzi,
ele,dekatDist);
        try
           eval(load_hrtf);
        catch
           fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',dekatDist,ele,jauhAzi,ele
,dekatDist);
           fail = true;
        end
        if fail == true
           fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
           continue;
        end
        b_l = cele*hrtf_mph_l;
        b_r = cele*hrtf_mph_r;

        load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',jauhDist,ele,dekatAzi,
ele,jauhDist);
        try
           eval(load_hrtf);
        catch
           fprintf('failed load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat\n',jauhDist,ele,dekatAzi,ele
,jauhDist);
           fail = true;
        end
        if fail == true
           fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
           continue;
        end
        c_l = (cazi)*hrtf_mph_l;
        c_r = (cazi)*hrtf_mph_r;


        interp_hrtf_l = a_l + b_l + c_l;
        interp_hrtf_r = a_r + b_r + c_r;
        if fail == false
```

```
            save_hrtf = sprintf('save
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_interptri2.mat interp_hrtf_l
interp_hrtf_r',dist,ele,azi,ele,dist);
            eval(save_hrtf);
        end
        fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
    end
  end
end
```

Source Code : make_mse_hrir_mph_rect.m

```
%interprect : using azi and ele
%interprect1: using dist and ele
%interprect2: using dist and azi

load coordinates_interp2d.mat

%azimuth and elevation
for dist = Distances
  elei = 1;
  for ele = Elevations2
    elei = elei+1;
    for azi = AzimuthStep(elei):AzimuthStep(elei):359
      load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele,azi,ele,dist);
      eval(load_hrir);
      load_hrir_interp = sprintf('load
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect.mat',dist,ele,azi,ele,di
st);
      fail = false;
      try
        eval(load_hrir_interp);
      catch
        fprintf('failed load
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect.mat\n',dist,ele,azi,ele,
dist);
        fail = true;
      end
      if fail == true
        fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
        continue;
      end
      if fail == false
        mse_l = 100*norm(hrir_mph_l-
interp_hrir_l)^2/(norm(hrir_mph_l)^2);
```

```
            mse_r = 100*norm(hrir_mph_r-
interp_hrir_r)^2/(norm(hrir_mph_r)^2);
            save_sd = sprintf('save
dist%d\\elev%d\\mse_hrir_mph_rect\\mse_hrir_mph_azi%d_elev%d_dist%d_
interprect.mat mse_l mse_r',dist,ele,azi,ele,dist);
            eval(save_sd);
         end
         fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
      end
   end
end

%distance and elevation
for dist = Distances
   elei = 1;
   for ele = Elevations2
      elei = elei+1;
      for azi = AzimuthStep(elei):AzimuthStep(elei):359
         load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele,azi,ele,dist);
         eval(load_hrir);
         load_hrir_interp = sprintf('load
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect1.mat',dist,ele,azi,ele,d
ist);
         fail = false;
         try
            eval(load_hrir_interp);
         catch
            fprintf('failed load
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect1.mat\n',dist,ele,azi,el
e,dist);
            fail = true;
         end
         if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect1.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
         end
         if fail == false
            mse_l = 100*norm(hrir_mph_l-
interp_hrir_l)^2/(norm(hrir_mph_l)^2);
            mse_r = 100*norm(hrir_mph_r-
interp_hrir_r)^2/(norm(hrir_mph_r)^2);
```

```
            save_sd = sprintf('save
dist%d\\elev%d\\mse_hrir_mph_rect\\mse_hrir_mph_azi%d_elev%d_dist%d_
interprect1.mat mse_l mse_r',dist,ele,azi,ele,dist);
            eval(save_sd);
         end
         fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
      end
   end
end

%distance and azimuth
for dist = Distances
   elei = 1;
   for ele = Elevations2
      elei = elei+1;
      for azi = AzimuthStep(elei):AzimuthStep(elei):359
         load_hrir = sprintf('load
dist%d\\elev%d\\azi%d_elev%d_dist%d_mph.mat',dist,ele,azi,ele,dist);
         eval(load_hrir);
         load_hrir_interp = sprintf('load
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect2.mat',dist,ele,azi,ele,d
ist);
         fail = false;
         try
            eval(load_hrir_interp);
         catch
            fprintf('failed load
dist%d\\elev%d\\hrir_azi%d_elev%d_dist%d_interprect2.mat\n',dist,ele,azi,el
e,dist);
            fail = true;
         end
         if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect2.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
         end
         if fail == false
            mse_l = 100*norm(hrir_mph_l-
interp_hrir_l)^2/(norm(hrir_mph_l)^2);
            mse_r = 100*norm(hrir_mph_r-
interp_hrir_r)^2/(norm(hrir_mph_r)^2);
            save_sd = sprintf('save
dist%d\\elev%d\\mse_hrir_mph_rect\\mse_hrir_mph_azi%d_elev%d_dist%d_
interprect2.mat mse_l mse_r',dist,ele,azi,ele,dist);
            eval(save_sd);
```

```
            end
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
        end
    end
end
```

Source Code : make_sd_interprect_hrtf.m

```
%interprect : using azi and ele
%interprect1: using dist and ele
%interprect2: using dist and azi
load coordinates_interp2d.mat

%azimuth and elevation
for dist = Distances
    elei = 1;
    for ele = Elevations2
        elei = elei+1;
        for azi = AzimuthStep(elei):AzimuthStep(elei):359
            load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dist,ele,azi,ele,dist);
            eval(load_hrtf);
            load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_interprect.mat',dist,ele,azi,ele,di
st);
            fail = false;
            try
                eval(load_hrtf);
            catch
                fprintf('failed load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat\n',dist,ele,azi,ele,dist);
                fail = true;
            end
            if fail == true
                fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect.mat
SKIPPED \n',dist,ele,azi,ele,dist);
                continue;
            end

            if fail == false
                sd_interp_l = norm(20*(log10(hrtf_mph_l)-
log10(interp_hrtf_l)))/sqrt(length(interp_hrtf_l));
                sd_interp_r = norm(20*(log10(hrtf_mph_r)-
log10(interp_hrtf_r)))/sqrt(length(interp_hrtf_r));
```

```
            save_hrtf = sprintf('save
dist%d\\elev%d\\sd_interprect_hrtf\\sd_interprect_hrtf_azi%d_elev%d_dist%d
_interprect.mat sd_interp_l sd_interp_r',dist,ele,azi,ele,dist);
            eval(save_hrtf);
         end
         fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
      end
   end
end

%distance and elevation
for dist = Distances
   elei = 1;
   for ele = Elevations2
      elei = elei+1;
      for azi = AzimuthStep(elei):AzimuthStep(elei):359
         load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dist,ele,azi,ele,dist);
         eval(load_hrtf);
         load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_interprect1.mat',dist,ele,azi,ele,d
ist);
         fail = false;
         try
            eval(load_hrtf);
         catch
            fprintf('failed load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat\n',dist,ele,azi,ele,dist);
            fail = true;
         end
         if fail == true
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect1.mat
SKIPPED \n',dist,ele,azi,ele,dist);
            continue;
         end

         if fail == false
            sd_interp_l = norm(20*(log10(hrtf_mph_l)-
log10(interp_hrtf_l)))/sqrt(length(interp_hrtf_l));
            sd_interp_r = norm(20*(log10(hrtf_mph_r)-
log10(interp_hrtf_r)))/sqrt(length(interp_hrtf_r));
            save_hrtf = sprintf('save
dist%d\\elev%d\\sd_interprect_hrtf\\sd_interprect1_hrtf_azi%d_elev%d_dist%
d_interprect.mat sd_interp_l sd_interp_r',dist,ele,azi,ele,dist);
            eval(save_hrtf);
```

```
            end
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
        end
    end
end
%distance and azimuth
for dist = Distances
    elei = 1;
    for ele = Elevations2
        elei = elei+1;
        for azi = AzimuthStep(elei):AzimuthStep(elei):359
            load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat',dist,ele,azi,ele,dist);
            eval(load_hrtf);
            load_hrtf = sprintf('load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_interprect2.mat',dist,ele,azi,ele,d
ist);
            fail = false;
            try
                eval(load_hrtf);
            catch
                fprintf('failed load
dist%d\\elev%d\\hrtf_azi%d_elev%d_dist%d_mph.mat\n',dist,ele,azi,ele,dist);
                fail = true;
            end
            if fail == true
                fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d_interprect2.mat
SKIPPED \n',dist,ele,azi,ele,dist);
                continue;
            end
            if fail == false
                sd_interp_l = norm(20*(log10(hrtf_mph_l)-
log10(interp_hrtf_l)))/sqrt(length(interp_hrtf_l));
                sd_interp_r = norm(20*(log10(hrtf_mph_r)-
log10(interp_hrtf_r)))/sqrt(length(interp_hrtf_r));
                save_hrtf = sprintf('save
dist%d\\elev%d\\sd_interprect_hrtf\\sd_interprect2_hrtf_azi%d_elev%d_dist%
d_interprect.mat sd_interp_l sd_interp_r',dist,ele,azi,ele,dist);
                eval(save_hrtf);
            end
            fprintf('dist%d elev%d hrtf_azi%d_elev%d_dist%d.mat DONE
\n',dist,ele,azi,ele,dist);
        end
    end
end
```