Spatial Audio Effects for Multichannel Loudspeaker Layouts

Yangang Cao

March 5, 2019

In the history of multichannel audio multiple different loudspeaker layouts with more than two loudspeakers have been specified. The most frequently used layouts are presented in this chapter. In the 1970s, the quadraphonic setup was proposed, in which four loudspeakers are placed evenly around the listener at azimuth angles $\pm 45^\circ$ and $\pm 135^\circ$. This layout was never successful because of problems related to reproduction techniques of that time, and because the layout itself had too few loudspeakers to provide good spatial quality in all directions around the listener.

For cinema sound, a system was evolved in which the frontal image stability of the standard stereophonic setup was enhanced by one extra center channel, and two surround channels were used to create atmospheric effects and room perception. This kind of setup was first used in Dolby's surround sound system for cinemas from 1976. Later, the layout was investigated, and ITU gave a recommendation about the layout in 1992. In the late 1990s, this layout also became common in domestic use. It is widely referred to as the 5.1 system, where 5 stands for the number of loudspeakers, and .1 stands for the low-frequency channel. In the recommendation, three frontal loudspeakers are at directions 0° and $\pm 30^\circ$, and two surround channels at $\pm(110 \pm 10)^\circ$. The system has been criticized for not delivering good directional quality anywhere but in front. To achieve better quality, it can be extended by adding loudspeakers. Layouts with 6–12 loudspeakers have been proposed.

In computer music, media installations and in academic projects, loudspeaker setups, in which the loudspeakers have equal spacing, have been used. In horizontal arrays the number of loudspeakers can be, for example, six (hexagonal array) or eight (octagonal array). In wave-field synthesis, the number of loudspeakers is typically between 20 and 200. In theaters and in virtual environment systems there exist systems in which loudspeakers are also placed above and/or below the listener.

2-D loudspeaker setups. In 2-D loudspeaker setups all loudspeakers are on the horizontal plane. Pair-wise amplitude panning is the best method to position virtual sources with such setups, when the number of loudspeakers is less than about 20. In pair-wise panning the sound signal is applied only to two adjacent loudspeakers of the loudspeaker setup at one time. The pair between which the panning direction lies is selected. Different formulations for pair-wise panning are Chowning's law, which is not based on any psychoacoustic criteria, or 2-D vector-base amplitude panning (VBAP), which is a generalization of the tangent law for stereophonic panning.

In VBAP a loudspeaker pair is specified with two vectors. The unit-length vectors $\mathbf{l}_m$ and $\mathbf{l}_n$ point from the listening position to the loudspeakers. The intended direction of the virtual source (panning direction) is presented with a unit-length vector $\mathbf{p}$. Vector $\mathbf{p}$ is expressed as a linear weighted sum of the loudspeaker vectors

$$\mathbf{p} = g_m \mathbf{l}_m + g_n \mathbf{l}_n.$$

Here g_m and g_n are the gain factors of the respective loudspeakers. The gain factors can be solved as

$$\mathbf{g} = \mathbf{p}^T \mathbf{L}_{mn}^{-1},$$

where $\mathbf{g} = [g_m \ g_n]^T$ and $\mathbf{L}_{mn} = [\mathbf{l}_m \ \mathbf{l}_n]^T$. The calculated factors are used in amplitude panning as gain factors of the signals applied to respective loudspeakers after suitable normalization, e.g., $||g||= 1$.

The directional quality achieved with pair-wise panning was studied. When the loudspeakers are symmetrically placed on the left and right of the listener, VBAP estimates the perceived angle accurately. When the loudspeaker pair is not symmetrical with the median plane, the perceived direction is biased towards the median plane, which can be more or less compensated.

When there is a loudspeaker in the panning direction, the virtual source is sharp, but when panned between loudspeakers, the binaural cues are unnatural to some degree. This means that the directional perception of the virtual source varies with panning direction, which can be compensated by always applying sound to more than one loudspeaker. As in pair-wise panning, outside the best listening position the perceived direction collapses to the nearest loudspeaker producing a specific sound. This implies that the maximal directional error is of the same order of magnitude with the angular separation of loudspeakers from the listener's viewpoint in pair-wise panning. In practice, when the number of loudspeakers exceeds about eight, the virtual sources are perceived to be in the similar directions in a large listening area.

A basic implementation of 2-D VBAP is included here as a **MATLAB** function.

```
function [gains] = VBAP2(pan_dir)

% Computes 2D VBAP gains for horizontal loudspeaker setup.
% Loudspeaker directions in clockwise or counterclockwise order.
% Change these numbers to match with your system.

ls_dirs=[30 -30 -90 -150 150 90];
ls_num=length(ls_dirs);
ls_dirs=[ls_dirs ls_dirs(1)]/180*pi;
% Panning direction in cartesian coordinates.
panvec=[cos(pan_dir/180*pi) sin(pan_dir/180*pi)];
for i=1:ls_num
% Compute inverse of loudspeaker base matrix.
    lsmat=inv([[cos(ls_dirs(i)) sin(ls_dirs(i))];...
    [cos(ls_dirs(i+1)) sin(ls_dirs(i+1))]]);
    % Compute unnormalized gains
    tempg=panvec*lsmat;
    % If gains nonnegative, normalize the gains and stop
    if min(tempg) > -0.001
        g=zeros(1,ls_num);
        g([i mod(i,ls_num)+1])=tempg;
        gains=g/sqrt(sum(g.^2));
```

```
23          return
24      end
25  end
```

3-D loudspeaker setups. A 3-D loudspeaker setup denotes here a setup in which all loudspeakers are not in the same plane as the listener. Typically this means that there are some elevated and/or lowered loudspeakers added to a horizontal loudspeaker setup. Triplet-wise panning can be used in such setups. In it, a sound signal is applied to a maximum of three loudspeakers at one time that form a triangle from the listener's viewpoint. If more than three loudspeakers are available, the setup is divided into triangles, one of which is used in the panning of a single virtual source at one time. 3-D vector-base amplitude panning (3-D VBAP) is a method to formulate such setups. It is formulated in an equivalent way to pair-wise panning in the previous section. However, the number of gain factors and loudspeakers is naturally three in the equations. The angle between the median plane and virtual source is estimated correctly with VBAP in most cases, as in pair-wise panning. However, the perceived elevation of a sound source is individual to each subject.

A basic implementation of 3-D VBAP for the loudspeaker setup as a **MATLAB** function.

```
1  function [gains] = VBAP3(pan_dir)
2  % Computes 3D VBAP gains for loudspeaker setup shown in Fig.6.4
3  % Change the lousdpeaker directions to match with your system,
4  % the directions are defined as azimuth elevation; pairs
5
6  loudspeakers=[0 0; 50 0; 130 0; -130 0; -50 0;  40 45; 180 45;-40 ...
       45];
7  ls_num=size(loudspeakers,1);
8  % Select the triangles of from the loudspeakers here
9  ls_triangles=[1 2 6; 2 3 6; 3 4 7; 4 5 8; 5 1 8; 1 6 8;
10 3 6 7; 4 7 8; 6 7 8];
11 % Go through all triangles
12 for tripl=1:size(ls_triangles,1)
13     ls_tripl=loudspeakers(ls_triangles(tripl,:),:);
14     % Panning direction in cartesian coordinates
15     cosE=cos(pan_dir(2)/180*pi);
16     panvec(1:2)=[cos(pan_dir(1)/180*pi)*cosE ...
           sin(pan_dir(1)/180*pi)*cosE];
17     panvec(3)=sin(pan_dir(2)/180*pi);
18     % Loudspeaker base matrix for current triangle.
19     for i=1:3
20         cosE=cos(ls_tripl(i,2)/180*pi);
21         lsmat(i,1:2)=[cos(ls_tripl(i,1)/180*pi)*cosE...
22         sin(ls_tripl(i,1)/180*pi)*cosE];
23         lsmat(i,3)=sin(ls_tripl(i,2)/180*pi);
24     end
25     tempg=panvec*inv(lsmat); % Gain factors for current triangle.
26     % If gains nonnegative, normalize g and stop computation
27     if min(tempg) > -0.01
28         tempg=tempg/sqrt(sum(tempg.^2));
29         gains=zeros(1,ls_num);
30         gains(1,ls_triangles(tripl,:))=tempg;
31         return
32     end
33 end
```