AN ALTERNATIVE IMPLEMENTATION OF VBAP WITH GRAPHICAL INTERFACE FOR SOUND MOTION DESIGN

Hongchan Choi

Stanford University
Center for Computer Research in Music and Acoustics (CCRMA)
660 Lomita Dr, Stanford, California, USA
hongchan@ccrma.stanford.edu

ABSTRACT

An implementation of vector-based amplitude panning (VBAP) for spatial display of sonified data is presented. The proposed method offers an implicit conversion from Spherical to Cartesian coordinates thus being particularly well suited for auditory display. Two techniques from computer graphics are adapted in order to predefine an optimum set of speaker triplets and perform the amplitude panning in real-time. Furthermore, the consideration of time delay from a virtual sound source to actual speakers is incorporated. Due to the geometrical nature of this procedure, the resulting system can be easily visualized by the graphic library OpenGL. Using this library I provide users with an intuitive control interface. A prototype is demonstrated that enables a user to compose a trajectory of sound in three dimensional space.

1. MOTIVATION

1.1. VBAP: Vector-Based Amplitude Panning

Vector-based amplitude panning (VBAP) [1] is one of several non-standard methods used to render virtual sound sources in 3D sound field using multiple speakers. VBAP is distinct in its clustering of adjacent speakers into triplets in which individual gain factors are calculated for each speaker in order to translate the sound into perceptually compelling spatial auditory cues.

The conventional VBAP method includes the following steps:

a) Define speaker triplets.

b) Position a virtual sound source (a new vector P) in the space.

c) Select a triangle (a speaker triplet) intersected by a vector between a sound source (P) and the position of listener (L).

d) Calculate 3 gain factors from each speaker on the triplet.

e) Interpolate gain factors from previous ones to new ones.

f) Iterate through steps b - e as needed.

Although a few implementations of VBAP have been adapted since the method's introduction in 1997 [2] [3], the procedure described here substitutes steps a), c) and d) with techniques from computer graphics. A novel approach emerges with a more intuitive visual interface and enhanced computational efficiency. The prototype transforms a spherical system (ambisonics and VBAP) into the Cartesian coordinate system, the one used by standard graphic libraries. This transformation bears a number of significant advantages including:

- integration with conventional graphic libraries, such as openGL and 3D vector calculation,
- a mapping paradigm that integrates well with data visualizations,
- an intuitive means of positioning and moving sound in virtual 3D space.

We describe Implementation of two algorithms adapted from 3D graphics, Quickhull [4] and Ray-Triangle intersection [5] following a description of *Field 8* a multi-touch interface for 2D panning.

1.2. *Field 8*, a multi-touch interface for 2D panning based on DBAP

Field 8 is a control user interface designed for distance based amplitude panning (DBAP) [6]. It provides an intuitive control interface on a multi-touch screen implemented on the iPad . Unlike other user interfaces of sound spatialization for VBAP or Ambisonics [7], the real-time user interaction and the rich visual feedback are focal points of the interface that allows users to draw multiple paths of sound motion with up to eight fingers. The site-specific prototype design for the CCRMA listening room [8], encompassing the user interface on iPad and a spatialization server built with ChucK audio programming language [9],provided a suitable environment to explore the sonic space in a 2D plane created by 8 speakers at ear level. The prototype has been successfully used in various performance contexts and compositions. *Field 8* is also useful for exploration and rapid design of appropriate scaling and mapping methods for auditory display. Expanding this potential to 3D auditory display using more than 8 speakers is clearly the next step.

1.3. Considering Efficiency and Usability

Adapting *Field 8* to 3D space using more than 8 speakers presents a number of logistical problems. The algorithm for DBAP should be redesigned to update an arbitrary number of gain controllers (in the site specific case cited here, 22 gain controllers) every few milliseconds. Deploying multiple sound sources will multiply the number of gain controllers. For example, a DBAP system requires to update 176 gain factors for every sample when there are 8 sound sources in motion. Furthermore, the system must calculate 176 distances between 3D points to get each gain factor meaning that the process involves 176 square-root operations in every iteration. A more efficient approach was needed to build a real-time spatialization system capable of handling multiple user interaction. The

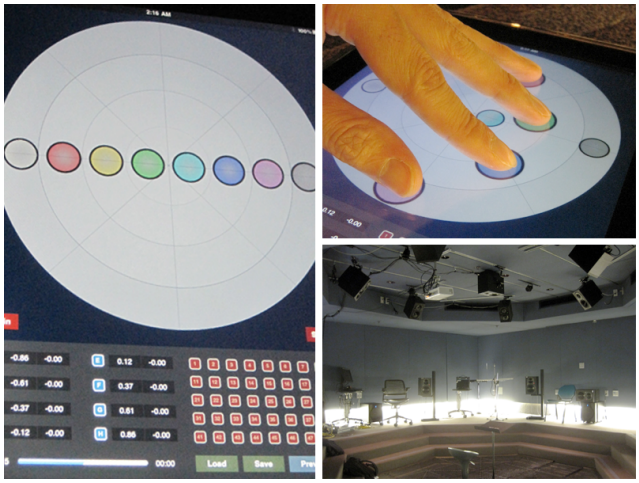


Figure 1: User interface of *Field 8* and CCRMA listening room

VBAP concept of grouping three proximate speakers proved a viable option which achieved a decent level of interactivity.

To summarize, the motivation for this work is twofold: developing a new panning system that

- can move multiple sound sources in a highly efficient fashion, and
- enables the user to design sound motions in an intuitive and expressive way.

2. SYSTEM DESIGN

2.1. Phase 1: Triangulation of Speakers

The operation of VBAP includes two separate procedures; the first phase is organizing the physical position of speakers in the space [2]. This procedure obtains an optimum set of non-overlapping speaker triplets(triangles), thus reducing the computational load by dealing with only 3 speakers at any given moment to calculate the panning. Unless the physical configuration of speakers is changed, the first phase needs to be updated only once.

Triangulating contiguous speakers in 3D space can be done in several different ways. Although the original source code uses triangulation [10] there was no specification in the original VBAP algorithm and subsequent papers of how it was implemented. My approach on this grouping task is to use a convex hull algorithm called Quickhull3D [4]. The convex hull of a set of points is the smallest convex set that contains the points. The algorithm originated from the field of computer graphics and is widely used to build a 3D mesh with a minimum set of triangles from an arbitrary number of vertices. (See figure 2.)

The Quickhull algorithm is highly optimized, so inserting a new vertex into the existing 3D mesh to build a new set of triangles on the fly is possible. Considering the largest speaker system in the world is using less than 200 speakers and the Quickhull algorithm can handle more than 200 vertices in real-time fashion, this algorithm is a viable way to triangulate speakers.

2.2. Phase 2: Ray-Triangle Intersection

The original VBAP algorithm calculates gain factors from a selected triplet by matrix operation. However, another method from computer graphics can be deployed to get gain factors. The Ray-Triangle intersection algorithm [5] is a 3D vector operation to calculate not only intersection of a ray vector and a triangle, but also the point of intersection. (See figure 3.)

If the position of a virtual sound source exists as a 3D point in the space, than we can assume a vector from the point of origin to the point of the sound source. If the physical configuration of speakers is a spherical mesh of triangulated speakers, the infinite extension of this vector intersects only one triangle (speaker group). This is particularly useful for VBAP operation because the Ray-Triangle intersection algorithm can infer an intersection point on a triangle, and the distances between 3 speakers of the triangle and this point can be calculated.

2.3. Relative Loudness and Time Delay

The Ray-Triangle intersection algorithm yields useful parameters. Rendering a realistic 3D auditory display is possible by using the loudness ratio between 3 speakers as well as the time delay estimated from the distance between a sound source and the speakers. Parameters from the algorithm are listed here (see also Figure 4).

a) A gain factor estimated from the distance between a sound source and the listener: P-L (in figure 4-(a))

b) 3 loudness ratios from distances between 3 speakers and a intersection point: S0-I, S1-I, S2-I (in figure 4-(b))

c) 3 delay times estimated from distances between 3 speakers and a sound source: S0-P, S1-P, S2-P (in figure 4-(c))

The system yields gain factors for 3 speakers by summing all 3 distances and dividing each distance by the sum as described in b). For example, when the intersection point moves to the exact same

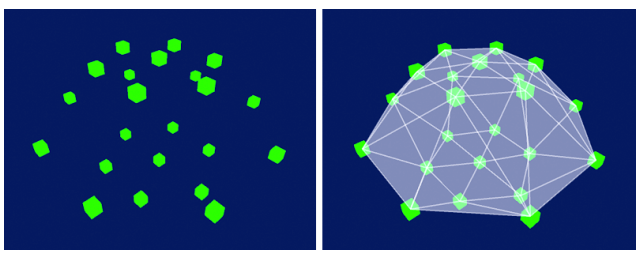


Figure 2: Using Quickhull algorithm to triangulate speakers

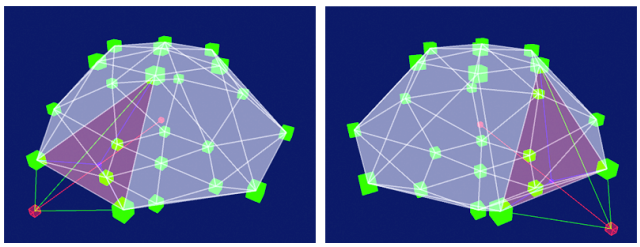
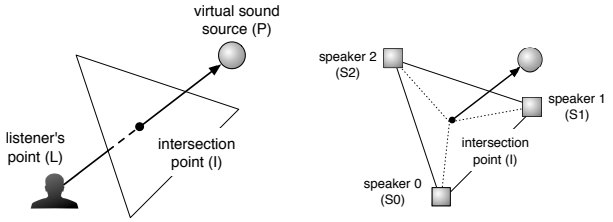
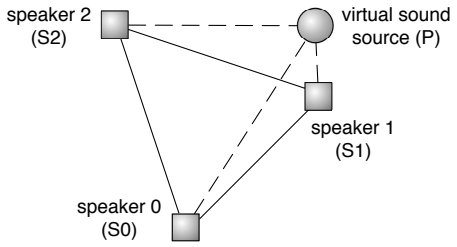


Figure 3: Ray-Triangle Intersection algorithm to select a triplet



(a) intersection point

(b) distance from 3 speakers



(c) distance from source

Figure 4: various relationships from intersecting algorithm

location as one of the 3 speakers, the gain factor for the speaker becomes 1.0 reducing the gain factors of the other two speakers to zero. In most cases, the intersection point moves from one triangle to another by passing through their shared edge. When the intersection point is located precisely on the edge, the sum of relative loudness of the two speakers on that edge will be 1.0. This will ensure the seamless transition when the intersection point moves across two triangles. This is partly similar with the DBAP method; however, it differs in its use of 3 speakers at any given moment. Also the distance between the listener and the sound source affects the overall loudness of the sound.

The original implementation of VBAP lacks the notion of time delay between a virtual sound source to selected speakers. Simply by calculating distances between a sound source from speakers in a selected group, the system can simulate time delay introducing the subtle change of timbre that arises from phase differences. Such concepts are integral to Wave field Synthesis.

Here we encounter a common obstacle in artificial spatialization methods: When the position of a virtual sound source is inside of a sound field the time delay of speakers will be a negative value, which is impossible in the real world, causing ambiguity in localization of the sound. Thus, this problem still remains in the system.

3. IMPLEMENTATION OF PROTOTYPE

A prototype built with two programming languages, Processing and ChucK, is demonstrated to test the feasibility and possible enhancements. Processing [11] functions as a core system that calculates the entire panning process and sends the result to ChucK [12] via an OSC(OpenSound Control) [13] connection. The VBAP object in ChucK renders a sound source in the space according to the data from a speaker triplet delivered from Processing. In this section, I describe design choices and details on implementation.

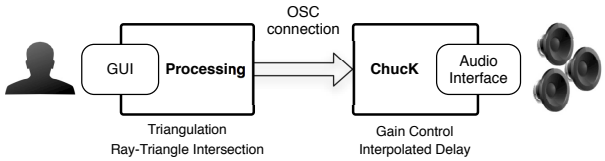


Figure 5: 2-Tier structure: Processing and ChucK via OSC

3.1. Processing: Spatialization Engine and User Interface

Processing is a visual programming language built for media arts and the classroom setting. It is widely used for designing prototypes or creating visual arts. One of its benefits is a large set of libraries that can be deployed with minimum effort. It is especially helpful to visualize data structures for better understanding.

The prototype implements Processing mainly because it has a nice library of vector calculation and a built-in OpenGL support. It significantly cuts the development time. A visualization is inherently correlated with a graphical user interface; thus having a compelling visualization is a clear advantage in terms of user control.

Since actual positions of speakers were initially in a spherical coordinate system, typical of a spatial audio setting, converting them into Cartesian coordinate system was required. This can be achieved when we understand that Zenith in a normal spherical system and Elevation in spatial audio are two orthogonal descriptions a vertical angle. This conversion into a Cartesian system enables us to perform a vector operation, a great advantage in terms of not only visualizing or animating what is happening, but also calculating required values from geometry algorithms since standard graphic libraries are based on the Cartesian system.

The system reads the speaker position data, converts them into Cartesian values, and then performs the quickHull3D algorithm to get an optimum set of speaker triplets. Convex hulling is a type of triangulation algorithm, that differs from the original triangulation algorithm in VBAP. In my implementation, this step is done by a library called newHull, a ported library from the original Java

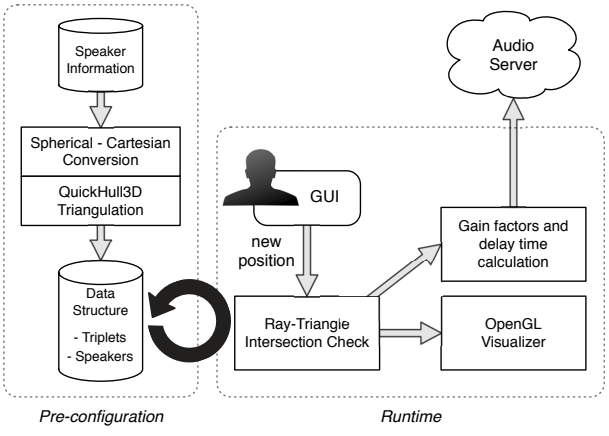


Figure 6: Sptialization Engine in Processing

implementation of quickHull3D.

The quickHull object in Processing is designed to produce a set of vertices(speakers), face indices (triangles, speaker triplets).

As a next step, the Ray-Triangle intersection algorithm checks all the triplets with a vector between the origin(listener) and a sound source. (See figure 4 (a).) The function that contains The Ray-Triangle intersection algorithm is the core of the whole system. It performs not only the essential visualization (lines between selected speakers and a sound source) but also calculates all the gain factors with delay times and sends OSC messages to the audio server. The OSC message consists of 3 parameters: an ID of a speaker, a gain factor(zero to one), and a delay time in milliseconds.

Moving a mouse can control the position of sound source. The camera will gradually follow the position of the sound source as it moves. Unlike other conversions made in the system, the movement of the mouse in a 2D plane can be converted into a 2 angle (Zenith, Azimuth) spherical coordinate system.

3.2. ChucK: Multi-Channel Audio Server

ChucK is a general-purpose programming language tailored for computer music. [12] miniAudicle, the front-end of the ChucK virtual machine, accelerated the prototype design process supporting concise programming and rapid experimentations. [14] A multichannel audio server is implemented in the ChucK language with miniAudicle. This server features 22 channels of audio to represent one or multiple sound sources in this iteration of the prototype. As mentioned, this prototype was designed for the CCRMA listening room. The site-specific details of this implementation are described in the next section.

The OSC data stream is dispatched to a respective speaker by the ID field. Note that the number of OSC packets is constantly 3 per speaker triplet and the Processing OSC sender will send these 9 numbers at every frame (about 16.7 milliseconds), 540 numbers per second. This is 7.3 times better than sending OSC packets for the entire set of 22 speakers with 3960 numbers for every second. For example, Field 8 was designed to control 8 speakers with 1440 OSC packets per second and occasionally encountered a bandwidth problem.

This is a significant advantage when detaching the control interface into a wireless device such as an iPad allowing users to control positions of sound sources without sitting in front of a workstation. The OSC packet size is consistently 9 numbers regardless of the number of speakers to be controlled and this consistency is possible thanks to the pre-configuration process of VBAP.

The interpolation between successive gain factors is performed by the built-in features of the Envelope objects in ChucK. The duration of interpolation is 16.7ms corresponding to the data speed from Processing rendering a seamless transition from previous gain factors to next ones. Unit generators for delay(DelayA) are interpolating delay times by default, so changing delay time does not introduce discontinuity in samples.

4. THE CCRMA LISTENING ROOM: SITE SPECIFIC SETUP

The CCRMA listening room is an experimental 3D space with 22 speakers and near-anechoic acoustics. The default 3D panning scheme is 3rd order Ambisonics (3v3h) that utilizes 16 channels of encoded audio streams. The OpenMixer [15] is a highly flexible software mixer running on the workstation. It transforms these encoded 16 streams into 22 audio channels routing the 22 speakers distributed in a sphere around the listener. The panning operation is accessible through a few experimental panners in Ardour [16], PureData [17] and SuperCollider [18].

As previously discussed, the prototype is tuned for the setting of the CCRMA listening room. However, it does not mean that the speaker position data is hard-coded in the software. Unlike the Ambisonics implementation, the new VBAP implementation performs triangulation on the fly from a text file with the positional information (either Spherical or Cartesian format). Therefore, it can be easily adapted to other venues without redesigning an encoder or a decoder.

The OpenMixer provides highly flexible audio input arrays including netJack [19] or JackTrip [20]. The ChucK audio server running on the laptop (MacBook Pro with a dual core CPU at

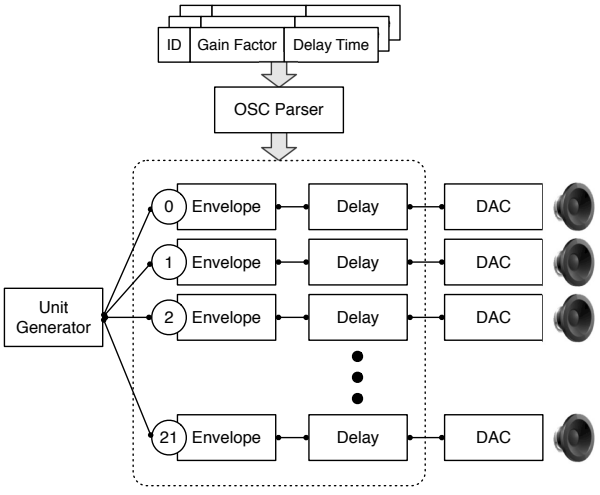


Figure 7: Multi-Channel Audio Server in ChucK

Figure 8: The CCRMA Listening Room

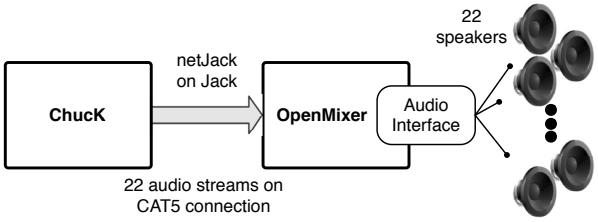


Figure 9: Connection between Audio Server and OpenMixer

2.2Ghz) could transfer 22 audio streams through the netJack driver without any problem.

5. CONCLUSION AND FUTURE WORK

In this study, I investigated a new VBAP implementation adopting two techniques from computer graphics to improve several aspects. The prototype presented the more intuitive and expressive graphical control interface. The embedded transformation of coordinate system creates synergies by tapping computer graphics libraries resulting in a highly responsive control interface.

However, the early implementation of the application has limitations. It does not have a sequencing feature yet, so it is not possible to record the trajectory of the sound. The system handles the sound material as a sound entity, rather than under the framework of "audio track" like typical sequencers or digital audio workstations. As of now, this prototype features only real-time interaction.

Future goals include a more sophisticated graphical user interface for 3D panning. This interface will include wireless and touchscreen devices for portability. To facilitate the calibration of varying speaker setups, I foresee using computer vision or 3d camera technology to quickly convey measurements to the system. Greater efficiency might be achieved if we integrate the panning operation into Chuck as a built-in unit generator. To mitigate the challenges of diversified setups, we could have a standardized method for notating speaker configurations. For example, the system could be calibrated for a given space by downloading an XML file from the website.

6. ACKNOWLEDGMENT

I would like to thank John Granzow and Jonathan Berger who provided invaluable suggestions and advice, to Fernando Lopez-Lezcano, for sharing his expertise in spatial audio and the CCRMA listening room. I am also grateful to Jarosalw Kapuscinski who provided with artistic and creative directions for the *Field 8* application and to Chris Chafe and Ge Wang whose related work inspired this research.

7. REFERENCES

[1] V. Pulkki, "Virtual sound source positioning using vector base amplitude panning," *Journal of the Audio Engineering Society*, vol. 45, no. 6, pp. 456–466, 1997.

[2] V. Pulkki and T. Lokki, "Creating auditory displays to multiple loudspeakers using vbap: A case study with diva project," in *International Conference on Auditory Display*, 1998.

[3] V. Pulkki, "Generic panning tools for max/msp," in *Proceedings of International Computer Music Conference*, 2000, pp. 304–307.

[4] C. Barber, D. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Transactions on Mathematical Software (TOMS)*, vol. 22, no. 4, pp. 469–483, 1996.

[5] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *Journal of graphics tools*, vol. 2, no. 1, pp. 21–28, 1997.

[6] T. Lossius, P. Baltazar, and T. de la Hogue, "Dbap-distance-based amplitude panning," in *Proceedings of 2009 International Computer Music Conference, Montreal, Canada*, no. 1, 2009.

[7] M. Gerzon, "Ambisonics in multichannel broadcasting and video," *J. Audio Eng. Soc*, vol. 33, no. 11, pp. 859–871, 1985.

[8] F. Lopez-Lezcano and C. Wilkerson, "Ccrma studio report," in *Proceedings of the International Computer Music Conference*, 2007.

[9] G. Wang, P. Cook *et al.*, "Chuck: A concurrent, on-the-fly audio programming language," in *Proceedings of International Computer Music Conference*, 2003, pp. 219–226.

[10] V. Pulkki, "Tkk akustiikka/tkk acoustics laboratory/software," http://www.acoustics.hut.fi/software/, retrieved February 2012.

[11] C. Reas and B. Fry, "Processing: programming for the media arts," *AI & Society*, vol. 20, no. 4, pp. 526–538, 2006.

[12] G. Wang, "The chuck audio programming language." a strongly-timed and on-the-fly environ/mentality"," Ph.D. dissertation, Princeton University, 2009.

[13] M. Wright, A. Freed, and A. Momeni, "Opensound control: State of the art 2003," in *Proceedings of the 2003 conference on New interfaces for musical expression*. National University of Singapore, 2003, pp. 153–160.

[14] S. Salazar, G. Wang, and P. Cook, "miniaudicle and chuck shell: New interfaces for chuck development and performance," in *Proceedings of the 2006 International Computer Music Conference*, 2006, pp. 63–66.

[15] F. Lopez-Lezcano and J. Sadural, "Openmixer: a routing mixer for multichannel studios," in *Linux Audio Conference 2010*, UtrechT, The Netherlands, 5/2010 2010.

[16] P. Davis *et al.*, "Ardour: digital audio workstation," http://ardour.org/, retrieved February 2012.

[17] T. Musil, M. Noisternig, and R. Höldrich, "A library for realtime 3d binaural sound reproduction in pure data (pd)," in *Proc. 8th Int. Conference on Digital Audio Effects*, 2005.

[18] J. McCartney, "Rethinking the computer music language: Supercollider," *Computer Music Journal*, vol. 26, no. 4, pp. 61–68, 2002.

[19] A. Carôt, T. Hohn, and C. Werner, "Netjackremote music collaboration with electronic sequencers on the internet," in *Proceedings of the Linux Audio Conference*, 2009.

[20] J. Cáceres and C. Chafe, "Jacktrip: Under the hood of an engine for network audio," *Journal of New Music Research*, vol. 39, no. 3, pp. 183–187, 2010.