Spatial Auditory Feedback

Evan Murray
emurray49@gatech.edu
Georgia Institute of Technology
Atlanta, Georgia, USA

Figure 1: Setup of the audio rendering system

ABSTRACT

This article will explore how camera tracking of the human body can be used as an input to control parameters of a spatial audio synthesis system. Furthermore, the potential relationship between spatial auditory cues and improvement in joint matching tasks will be explored.

CCS CONCEPTS

• **Human-centered computing** → **Auditory feedback**; **Gestural input**; • **Computing methodologies** → Motion capture.

KEYWORDS

Ambisonics, Spatial Audio, Camera Tracking, Auditory Feedback

ACM Reference Format:
Evan Murray. 2024. Spatial Auditory Feedback. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Proprioception is a term describing the sense of the body which involves awareness of the limbs and joints [3]. Panoptic pose estimation is a computer vision based approach where "every pixel on the body is potentially a semantically meaningful keypoint that has an individual identity," [12]. Using these two concepts, the user can focus on practicing proprioceptive awareness while pose estimation observes the motion of the user as a control input. It is hypothesized users will have a lower mean squared error for ipsilateral joint matching tasks as the order of ambisonics increases because the brain will be able to relate the timbral information generated by the audio rendering system to the spatial information produced by ambisonic output.

2 THEORY

Various studies show auditory cues in the environment surrounding individuals have a significant impact on said individual's awareness of their body in space. One study has shown the improvement of pose matching with training via auditory and proprioceptive feedback [2]. Another has demonstrated the reduction of postural sway to be directly proportional to the number of sound sources surrounding the user in a sound field [5]. Others show the link between head movement and reduction in sound localization error in both the azimuth and vertical planes [7, 8]. These findings illustrate the multi-modality of the human learning process for certain tasks, such as the joint matching tasks described in the introduction. There is a significant link to improved learning and performance with an increase in modalities. In the examples above, the brain is able

to associate auditory cues with feedback from the somatosensory system—responsible for the internal muscle tension senses which influence proprioception. Many of the studies above have already determined a link between proprioception and auditory feedback. Thus, the goal of this study is to focus on the role spatial resolution plays in influencing the accuracy of the joint position matching task by varying the order of ambisonics. A control where all of the speakers play the same audio with only one timbral feature will be present for the null hypothesis.

3 TECHNOLOGY

3.1 Kinect Pose Estimation

Gathering human poses in accurately in 3D space was one of the greatest challenges of this project. The majority of the project timeline was spent researching which sensor was the most accessible yet accurate. RGB cameras with the absence of depth data are able to estimate human pose from 2D tensors using machine learning, as is done in [12]. Although some of these models may be trained on depth data, they tend to perform poorly in low-lighting conditions—as the RGB data gathered from a camera is not sufficient for the model to make an accurate inference in these conditions. An alternative approach is to use a pose estimation model which supports combined depth and RGB data for the input. Depth data can be obtained through various methods in low lighting conditions using infrared light or lasers. Apple has an interface available on iPhone and iPad which allows developers to fetch data from the LiDAR sensor and cameras [1]. This interface did not produce usable results, as the data returned from the observation included a great amount of jitter. Additionally, this software is not open source and only accessible to those who have the corresponding devices. Thus, the current system is implemented using the Xbox 360 Kinect which performs depth detection through its infared sensor and has a community dedicated to developing open source software, such as in the package SimpleOpenNI [9].

3.2 Processing

Processing is another open source program which makes prototyping and programming relatively simple and is often easier than building a fully-fledged application. The graphic functionality is handled in one single `draw()` function [4]. SimpleOpenNI includes an example which draws a skeleton on the user detected by the Kinect [2]. This code can be used as a visual interface displayed to the user while the program is running.

3.3 Open Sound Control

The detected joint positions and orientations are then sent to the machine running the ambisonic rendering system using Open Sound Control (OSC)—a networking protocol which allows musical control parameters to be sent as UDP messages associated with a path and IP address. There is a package available in Processing for sending OSC commands called oscP5 which is used in this project [11].

3.4 Code Snippets

To configure the Kinect and OSC server, the following initialization code is run in the `setup()` function of the Processing sketch. The

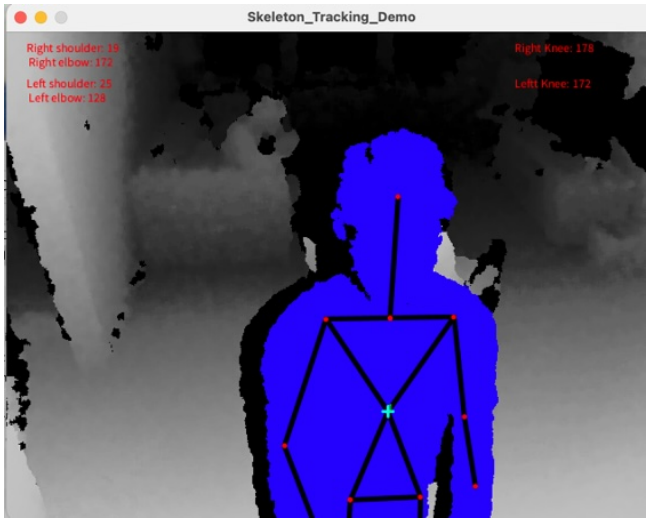

Figure 2: SimpleOpenNI with skeleton tracking demo in Processing

`jointOrientation` and `jointPos` are empty data structures which are initialized for storing the data retrieved by the Kinect sensor:

```
SimpleOpenNI kinect = new SimpleOpenNI(this);
OscP5 oscP5 = new OscP5(this, 6666);
myRemoteLocation = new NetAddress("10.11.12.100", 8888);
PMatrix3D jointOrientation = new PMatrix3D();
PVector jointPos = new PVector();
```

Listing 1: Initialization code

This single line of code runs inside of the `draw()` function and is responsible for fetching the active users detected by the Kinect:

```
kinect.getUsers(userList);
```

Listing 2: Active Kinect users

Next, the first detected user is selected, along with a joint type, to start tracking. The `sourceId` determines which source is influenced by the OSC messages in the audio rendering software:

```
int userId = userList.get(0);
int jointId = SimpleOpenNI.SKEL_LEFT_KNEE;
int sourceId = 4;
```

Listing 3: Tracking configuration

The same data which is used to draw the limbs on the detected user's skeleton can be fetched using each of the functions listed below. These functions return a confidence value which can be used to filter out values with lower confidence. To retrieve the detected values, the data structures defined in listing 1 are passed into the functions below and updated with the retrieved data. In this example, the azimuth is calculated in radians using members of a rotation matrix returned by the first method:

```
float rotationConfidence = kinect.
    getJointOrientationSkeleton(userId, jointID,
    jointOrientation);
float positionConfidence = kinect.
    getJointPositionSkeleton(userId, jointID, jointPos);
if (rotationConfidence < 0.2) { return; }
if (positionConfidence < 0.2) { return; }
```

```
float phi = atan2(jointOrientation.m20, jointOrientation.
    m21);
```

Listing 4: Fetching data

Once the data is fetched from the Kinect, the global variables can be appended to various OSC paths to send control messages to the ambisonic encoders for each sound source. The `/source/azim` path controls the azimuth of the source while the `/source/xyz` path specifies the Cartesian coordinates of the source in 3D space:

```
OscMessage myMessage1 = new OscMessage("/source/azim");
OscMessage myMessage2 = new OscMessage("/source/xyz");
myMessage1.add(sourceId);
myMessage1.add(phi);
myMessage2.add(sourceId);
myMessage2.add(jointPos);
```

Listing 5: Creating OSC Messages

Finally, the OSC messages can be sent through the `OscP5` object and received by the rendering software:

```
oscP5.send(myMessage1, myRemoteLocation);
oscP5.send(myMessage2, myRemoteLocation);
```

Listing 6: Sending OSC Messages

3.5 Rendering Software

The rendering setup uses a Linux-based machine with JACK (JACK Audio Connection Kit) installed for routing the capture inputs and the capture outputs. The capture device is a Klark Teknik DN9630 which converts USB audio to AES50 (over ethernet) signals sent to the speakers and the mixer within the lab. Pure Data is used to generate the sound source which is white noise connected to a down sampler and low-pass filter for timbral effects. To encode the sound source in Pure Data into Higher Order Ambisonics, this SuperCollider quark is used along with OSC listeners which influence the parameters passed into the module [6]. The encoded signal is then passed to the decoder which is the AllRADecoder developed by the Institute of Electronic Music and Acoustics [10]. The code for the full project is available in this repository: https://github.com/emurray2/SpatialAudio/tree/main/MUSI7100Research/spatial-auditory-feedback.

4 CONCLUSION

Although the semester timeline was short and didn't allow for a formal study, the project was presented at a demo day setting and the audience had a chance to interact with a prototype of the system. The initial prototype featured one posture which controls the timbre of the synthesis and audio rendering software discussed above. The user head position was mapped to the noise filter and the left knee was mapped to the azimuth of the HOA-SC module to control the horizontal azimuth of the rendered sound source. This worked well for having users experiment with doing squats and single leg squats with their knee oriented in a single direction. The lowpass filter signified reaching the lower position of a squat and when the user stood back up they would know they completed one rep of the posture, as the sound returned to the original parameters. The azimuth being rotated with the knee also encouraged users to explore more variations of the posture. One of the challenges was having the Kinect be able to detect the active user with other people in the background. This could be specific to the `SimpleOpenNI` module discussed above, however further exploration is needed to determine an easier way for the Kinect to recognize the active user. Another challenge was the parameter range did not work the same for users of different height. This is because it is currently hard-coded and future iterations of the project should include a dynamic way to detect the user's height and map the timbral parameters to the appropriate range.

REFERENCES

[1] Apple. 2023. *VNDetectHumanBodyPose3DRequest.* Apple Developer. https://developer.apple.com/documentation/vision/vndetecthumanbodypose3drequest

[2] Anna Vera Cuppone, Giulia Cappagli, and Monica Gori. 2018. Audio Feedback Associated With Body Movement Enhances Audio and Somatosensory Spatial Representation. *Frontiers in Integrative Neuroscience* 12 (Sept. 2018), 6. https://doi.org/10.3389/fnint.2018.00037 Publisher: Frontiers.

[3] Ellen Fridland. 2011. The case for proprioception. *Phenomenology and the Cognitive Sciences* 10, 4 (Dec. 2011), 521–540. https://doi.org/10.1007/s11097-011-9217-z

[4] Ben Fry, Casey Reas, and Dan Shiffman. 2001. *Processing 4.* Processing Foundation. https://github.com/benfry/processing4

[5] Lennie Gandemer, Gaetan Parseihian, Richard Kronland-Martinet, and Christophe Bourdin. 2017. Spatial Cues Provided by Sound Improve Postural Stabilization: Evidence of a Spatial Auditory Map? *Frontiers in Neuroscience* 11 (June 2017), 5. https://doi.org/10.3389/fnins.2017.00357 Publisher: Frontiers.

[6] Florian Grond. 2021. *SC-HOA.* Fonds de Recherche du Québec - Société et Culture (FRQSC). https://github.com/florian-grond/SC-HOA?tab=readme-ov-file

[7] Akio Honda, Hiroshi Shibata, Souta Hidaka, Jiro Gyoba, Yukio Iwaya, and Yôiti Suzuki. 2013. Effects of Head Movement and Proprioceptive Feedback in Training of Sound Localization. *i-Perception* 4, 4 (June 2013), 253–264. https://doi.org/10.1068/i0522 Publisher: SAGE Publications.

[8] Ken I. McAnally and Russell L. Martin. 2014. Sound localization with head movement: implications for 3-d audio displays. *Frontiers in Neuroscience* 8 (Aug. 2014), 5. https://doi.org/10.3389/fnins.2014.00210 Publisher: Frontiers.

[9] Max Rheiner. 2018. *SimpleOpenNI.* GitHub. https://github.com/totovr/SimpleOpenNI/tree/master

[10] Daniel Rudrich. 2021. *AllRADecoder.* Institute of Electronic Music and Acoustics. https://github.com/studiorack/iem-plugin-suite/tree/master/AllRADecoder

[11] Andreas Schlegel. 2015. oscP5: An OSC library for java and the programming environment Processing. https://doi.org/10.5281/zenodo.16308

[12] Shaokai Ye, Anastasiia Filippova, Jessy Lauer, Steffen Schneider, Maxime Vidal, Tian Qiu, Alexander Mathis, and Mackenzie Weygandt Mathis. 2023. SuperAnimal pretrained pose estimation models for behavioral analysis. https://doi.org/10.48550/arXiv.2203.07436 arXiv:2203.07436 [cs, q-bio].