Adaptive Beamforming Project in C++ and Pure Data

Xiaoxuan Zhang
xiz031@ucsd.edu
Supervised by Shahrokh Yadegari, Grady Kestler

Project Summary:

This summer, I devoted my time to the adaptive beamforming project working in a group of 4. Developing from the original beamforming algorithm from Elliot M. Patros [2], our group was able to upgrade the algorithm, making it adaptive for different user locations. Currently, we realized the function, in which the users can interact with different X, Y, Z axis inputs and perceive a real-time update of the generated beam of sound. From a backend perspective, our "beamforming∼" object in Pure Data takes in different X, Y, Z input and constantly generates a set of new filters to form the new beams for the original audio input in low latency, enabling the audience to feel the beam moving based on their needs in real-time. Despite the fact that we have completed the project, there are still possible future steps to upgrade the work. Adding a camera module and making the user location an automated input extracted by computer vision will be a great direction to move forward.

Personal Contributions:

1. Modified the partconv object (from Pd-Spectral-Toolkit [1]), making it convolve at real-time.
2. Created multiple patches used in 14-channel convolution of filters and input audio, automating the filter inputs, GUI for final presentation.
3. Execute modular testing with code from Pure Data, MATLAB and Max MSP. Test compilation of externals on MacOS.
4. Join weekly meetings actively, final presentation. Manage files and contribute to documentation.

Project Experience:

For the period of this project, Helen and I started at the end of the spring quarter, and we had Eduard and Gabriel joining us in the summer, working as a team, to be able to successfully reach our goal of making the beamforming algorithm adaptive. At the very beginning, Helen and I spent a lot of time figuring out the MATLAB scripts and app from the git repository and the local copy on one of our demo computers, trying to give ourselves a clearer and more comprehensive view by selecting the deterministic files that need to be translated into the other language and later implement into the audio programming software. During this stage, we attempted to translate the MATLAB files into Python, as the beamforming toolbox requires several large matrix calculations and contains interactive software, we assumed that the comprehensive matrix library and the graphic user interface designing library in Python will give us sufficient help in the project. When

Edward and Gabriel joined the project at the beginning of the summer, we agreed on switching to C++, in order to reach a fast speed in matrix calculation and result in a lower delay time, which Python couldn't perform as well. Therefore, we switched to translating from MATLAB to C++ and reassigned the responsibilities inside our group. Originally, we were using a combination of the filters generated by the MATLAB algorithm and Max MSP to realize the beamforming of the audio input, but now, since we want to make the "beamforming$\sim$" object adaptive based on the interactive input from the users, we decided to combine the GUI into the audio programming software and to use Pure Data instead of Max MSP we used earlier.

While working towards the final goal, the plan we made earlier is very helpful. Although we hesitated on some of the designs throughout the project and changed some details under the different functions we want to achieve, we were still able to be in pace, as two sub-groups are working in parallel and matching the general timeline closely. One topic that we spent a lot of time discussing was on how to output the generated filters from the beamforming object and taking them as the inputs to the convolution patch expecting 14-channel audio output. One is to have 28 outlets from the "beamforming$\sim$" object and connect them to a 14-inlet "convolution$\sim$" object, which contains 28 "partconv$\sim$" objects under the hood. Option two was to write down all the filters and read in them individually with "partconv$\sim$" object. Fortunately, both methods of implementation will not cause a significant delay time and we don't need to categorize time as one compound variable. In order to adapt to various sound systems in the future, we decided to use the later design, making our algorithm able to be easily implemented on different sets of speakers, for example, 12 channels and 16 channels instead of 14 channels.

During the entire project span, although everyone had different work as assigned in the weekly meetings, we still took a look at each other's code, analyzed how the signals were supposed to work on the whiteboard or helped others understand the libraries and documentation during the week frequently. Besides the major progress I have in signal processing and Pure Data programming, I also practiced git / version control and compilation of externals a lot. Regarding soft skills, I also feel more confident in exchanging coding ideas verbally and extracting important messages from the meetings. I look forward to joining future project groups as the academic year unveils.

References

[1] Cooper Baker. *Pd-Spectral-Toolkit*. 2013. URL: `https://github.com/cooperbaker/Pd-Spectral-Toolkit`.

[2] Elliot M. Patros. "Effort compression: signal-dependent gain management for the pressure matching beamforming method." PhD thesis. University of California, San Diego, 2019. URL: `https://www.proquest.com/docview/2348319109`.