PanoSterec.js

(AudioWorkletProcessor module for 2-D panoramic stereo recording)

version 1.0.0

User guide

Kaoru Ashihara

April 21, 2022

The JavaScript file 'PanoSterec.js' is the AudioWorketProcessor module for the 2-channel 2-dimensional panoramic recording. It makes you capable to capture the immersive 2-dimensional panoramic sound in the browser by a stereo (2-track) recording with either a stereo microhone system 'ZOOM iQ6' or a handy recorder such as 'ZOOM H6' and 'TASCAM DR-22WL.'

The browser has to support AudioWorklet.

1 Hardware preparation

1.1 ZOOM H6

If you use 'ZOOM H6' as the recording device, make sure the XY stereo microphone capsule 'ZOOM XYH-6' is attached to the device. Connect the device to your computer with a USB cable. Turn on the ZOOM H6, set it as an audio interface and select the stereo mix mode.

Since the stereo microphones of the ZOOM XYH-6 are asigned to the L and R tracks of the ZOOM H6, activate these tracks by pressing the L and R buttons on the front panel of the ZOOM H6. If the ZOOM H6 is selected as the audio input device of your computer, you are ready for the recording.

1.2 TASCAM DR-22WL

Since a stereo handy recorder 'TASCAM DR-22WL' cannot be used as a USB audio interface, you have to get a stereo analog audio signal from its phone-out jack and digitize it with a USB audio interface connected to your computer. Instead of the audio interface, you can use a USB audio capture cable as shown in Fig. 1.

The recorder has to be in the recording standby mode otherwise you cannot get any signal from its phone-out jack. Make sure if the USB audio is selected as the audio input device of the computer.

1.3 ZOOM iQ6

In the case of the stereo microphone system 'ZOOM iQ6,' a stereo analog audio signal has to be captured from its phone-out jack and digitized by either a USB audio interface or a USB audio capture cable that is connected to the computer. To supply power to and activate the ZOOM iQ6, it has to be attached to the iOS device such as iPhone and iPad and its sound recorder

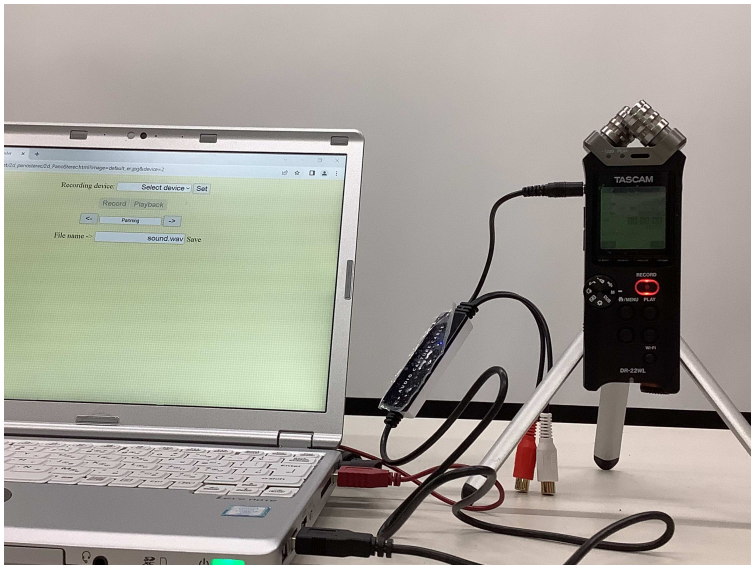

Fig. 1: Getting audio signal from TASCAM DR-22WL
When 'TASCAM DR-22WL' is in the recording standby mode, you can capture a stereo analog audio signal from the phone-out jack of the recorder. The signal can be digitized and sent to the computer by a USB audio capture.

app has to be running as can be seen in Fig. 2. The USB audio has to be selected as the audio input device of the computer.

2 Setting up AudioWorkletNode

To use 'PanoSterec.js' in your Web page, you have to add it as the AudioWorletProcessor to the AudioContext.audioworklet in the main thread script. Sample source code to do this is shown below.

code 1

```
await audioctx.audioWorklet.addModule('PanoSterec.js').then(()=⟩

        worklet = new (window.AudioWorkletNode ||
        window.webkitAudioWorkletNode)(audioctx, 'PanoSterec');

                                    .
                                    .
) .catch(console.error);
```

In the code above, 'audioctx' is the AudioContext instance and 'worklet'

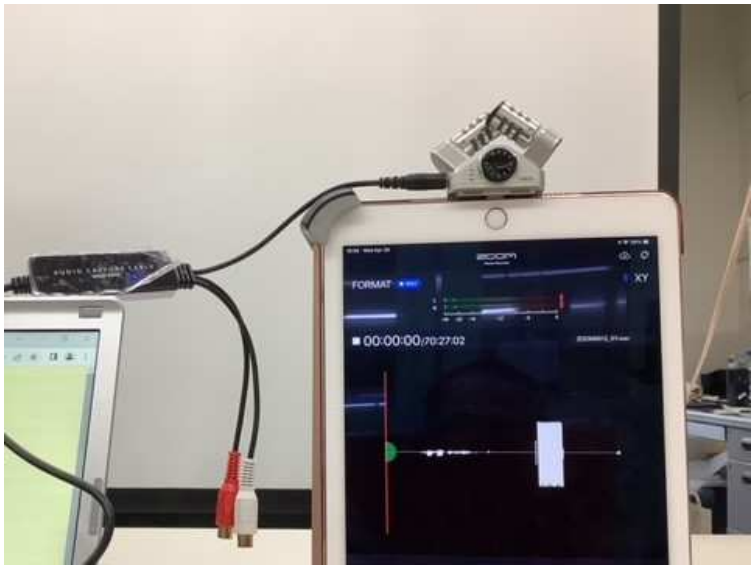

Fig. 2: Getting audio signal from ZOOM iQ6
In this example, since the ZOOM iQ6 is attached to the iPad on which the recording app is running and a USB audio capture cable is attached to the phone-out jack of the ZOOM iQ6, a stereo analog audio signal can be captured, digitized and transmitted to the computer.

is the AudioWorletNode instance.

Once the AudioWorkletProcessor is added to the AudioContext, you can send a message to the AudioWorkletProcessor by the following format.

```
worklet.port.postMessage({type : value});
```

Each message can consist of a title and body. In the code above, for instance, 'type' is the message title and 'value' is its body.

Before start recording the sound, let the AudioWorkletProcessor know the audio sample rate and which recording device is being used by posting a message to the AudioWorkletProcessor.

In 'PanoSterec.js,' the recording device is identified by an integer ID that is either 0, 1, or 2. 0 is for 'ZOOM H6.' 1 and 2 are for 'ZOOM iQ6' and 'TASCAM DR-22WL,' respectively. For example, if the sample rate is 44100 Hz and the recording device is 'TASCAM DR-22WL,' you can use the code below.

code 2

```
let samplerate = 44100;
let devID = 2;                          // 2 is for TASCAM DR-22WL
                                        .
                                        .
worklet.port.postMessage({"rate":samplerate, "device":devID});
```

The AudioWorkletProcessor can also send messages to the main thread. In the case of 'PanoSterec.js,' only 2 messages are posted by the Audioworklet-Processor. One of them is titled 'samples' and it conveys an integer number of the samples processed (per channel).

The other message conveys a stream of the processed data and has a title 'audio.' You can receive these messages from the AudioWorkletProcessor by the code below.

code 3

```
let currSamples;
let audioData = [];
                                        .
                                        .
worklet.port.onmessage = (event) =⟩{

      if(event.data.samples)

            currSamples = event.data.samples;

      if(event.data.audio)

            audioData.push(event.data.audio);

};
```

These messages are being sent each time the input buffer is full and the buffered data are processed. By receiving them constantly during the recording, the progress of the recording can be checked.

Code 1, code 2, and code 3 above can be put together as follows.

```
let samplerate = 44100;
let devID = 2;                          // 2 is for TASCAM DR-22WL
let currSamples = 0;
let audioData = [];
                                        .
```

```
                                        .
await audioctx.audioWorklet.addModule('PanoSterec.js').then(()=>{
        worklet = new (window.AudioWorkletNode ||
        window.webkitAudioWorkletNode)(audioctx, 'PanoSterec');
        worklet.port.postMessage({"rate":samplerate, "device":
        devID});
        worklet.port.onmessage = (event) =>{
                if(event.data.samples)
                        currSamples = event.data.samples;
                if(event.data.audio)
                        audioData.push(event.data.audio);
        };
}) .catch(console.error);
                                        .
                                        .
```

3 Recording

The following code makes the AudioWorkletProcessor get ready to capture data.

```
worklet.port.postMessage({"state":"run"});
```

The AudioWorkletProcessor does not know, however, where data come from. It is necessary, therefore, to connect the proper data source to the AudioWorkletNode. The data source in this case is the audio input device and to use the audio input device as the data source on the Web page, you have to get permission to use it. Sample code to do this is shown below.

```
let constraints = {
        audio: {
                mandatory: { echoCancellation: false,
                googEchoCancellation: false }, optional: []
```

```
        }, video: false

};
// Get permission to use media device
navigator.mediaDevices.getUserMedia(constraints).then(handleSuccess);
```

When navigator.mediaDevices.getUserMedia returns SUCCESS, you are allowed to use the audio input device and a function 'handleSuccess' is called. The code below is a sample of the function.

```
let handleSuccess = function(stream){

        media = audioctx.createMediaStreamSource(stream);
        // Connect source, processor, and destination
        media.connect(worklet);
        worklet.connect(audioctx.destination);

};
```

In this function, the source (recording device) is connected to the AudioWorkletNode and the AudioWorkletNode is then connected to the destination (audio output device). This allows the data captured by the recording device to be processed by the AudioWorkletProcessor and the processed data to be reproduced through the audio output device. To listen to the reproduced sound, stereo headphones have to be connected to the audio output device.

During the recording, you can stop it by posting a message to the AudioWorkletProcessor. Sample code to stop recording is shown below.

```
worklet.port.postMessage({"state":"played"});
// Disconnect
worklet.disconnect();
media.disconnect();
```

As can be seen in the code above, after quitting recording, the destination and the source have to be disconnected from the AudioWorkletNode.

Once the recording started, buffering and processing of incomming data are repeated until the recording is stopped. Each time the buffered data are

processed, the processed data are sent from the AudioWorkletProcessor to the main thread. As previously shown in code 3, code to collect the processed data in the main thread script is something like as follows.

```
let audioData = [];
                                        .
                                        .
worklet.port.onmessage = (event) =⟩{
        if(event.data.audio){
                audioData.push(event.data.audio);
        }
};
```

When the recording is done, the array 'audioData' contains the processed data stream and you can use it as a stream of the stereo linear PCM data.

4 Panning control

During the recording and reproduction, panning of the sound can be controlled by posting a message to the AudioWorkletProcessor. Sample code to do this is shown below.

```
let horizontalAngle;
                                        .
                                        .
worklet.port.postMessage({"pan":horizontalAngle});
```

In the code above, the variable 'horizontalAngle' is the angle (in degrees) in the horizontal plane. An integer between -180 and 179, inclusive, has to be asigned to it. The default (initial) value for panning in 'PanoSterec.js' is 0 degree.

5 Closing remarks

In this document, how to use 'PanoSterec.js' as the AudioWorkletProcessor module in your Web page is briefly described. By adding several lines to

your main thread script, 360-degrees of immersive panoramic sound can be experienced.

Please note that it uses only a common stereo audio format. Recording of spatial audio such as 5.1 surround sometimes requires many input channels. It is not easy, however, to use 3 or more audio input channels simultaneously in a Web page. Even if you have microphones and a multi-track audio interface, it is still difficult to handle multi-track recording in the browser.

By using 'PanoSterec.js,' a scene with panoramic sounds can be reproduced or created without difficulty. You may be able to combine the procedure with the panoramic video imaging technology to construct the whole panoramic scene on your Web site. Enjoy programming.

6 License of the program

MIT license