Audio Signal Processing Laboratory #3: Spatial Audio

Intro Video: https://youtu.be/WR41YlzAm0k

1

What to Submit:

- Your code and Simulink block diagrams. Make sure you are saving all Simulink models to your Current Folder in MATLAB
- A report with answers, figures, and explanations
- Any sound files that are asked for
- When to Hand It In:
  - It is due immediately at the beginning of the day (i.e., 0:00AM) specified on the course calendar. **Late assignments will receive a grade deduction of 20% of the full grade each day.**

2

How to Submit:

- Put all your files in one folder. Compress this folder and name it <firstname>_<lastname>_HW8.zip , ie “Zhiyao_Duan_HW8.zip”.
- Submit via blackboard to the appropriate entry. You can make multiple attempts but only the last attempt will be graded.
- **New to these labs:** You are provided scripts that allow you to demonstrate your real-time Simulink audio for each spatial audio algorithm. A sample demonstration is found here: https://youtu.be/rj6sgMC2I-g

3

Problem 1: Equal Power Panning

The first algorithm we will look at implements a simple equal power panning algorithm

Video Tutorial: https://youtu.be/eva0C5nMNfo

4

The Algorithm

To pan an input sample (**called 'x'**) using equal power panning, the output samples (**called y[left] or y[right]**) on the left and right is obtained by the following formula:

$$\mathrm{y[left]} = \mathrm{x}\cos\left(\lambda\,\frac{\pi}{2}\right),$$
$$\mathrm{y[right]} = \mathrm{x}\sin\left(\lambda\frac{\pi}{2}\right),$$

The parameter λ (or "lambda") ranges from 0 to 1 and controls how much signal energy is sent to the left channel and to the right channel.

5

Part A) Plot the Panning Curves (0.5 Points)

Plot the sine and cosine components of the equal power panning algorithm, ie, on the same set of axis (therefore, **not** using subplot), plot the following:

$$\cos\left(\lambda\,\frac{\pi}{2}\right),$$
$$\sin\left(\lambda\frac{\pi}{2}\right),$$

The x-axis should be the lambda parameter, ranging from 0 to 1. Use "linspace" to make a linearly spaced array of lambda from 0 to 1 over 1000 points, and then plot the above coefficients. Use "legend" to label the two curves.

6

Part B) Run Initialization (0 Points)

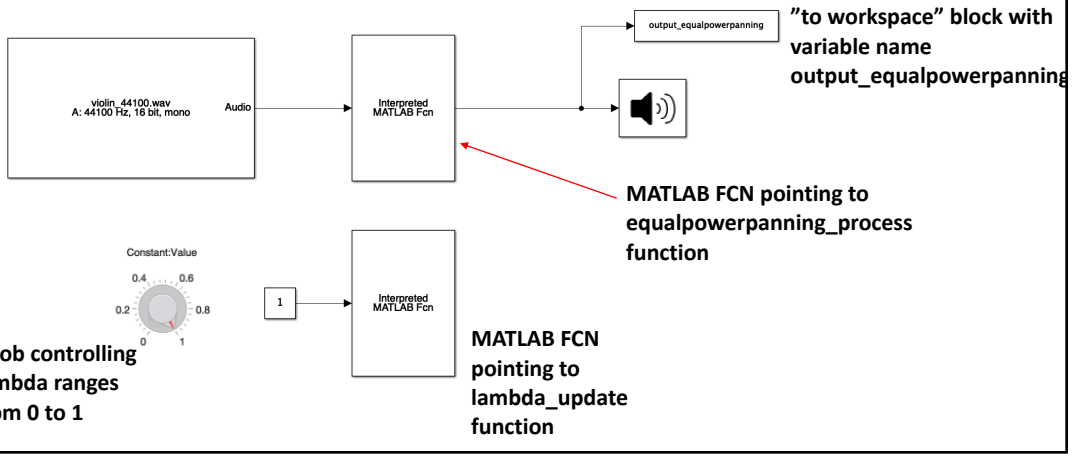
Look at the provided script *equalpowerpanning_init.m* in which, similarly to the previous labs, a global variable is defined to control the lambda parameter. You do not need to do anything to this script, but you **do need to run it in the MATLAB Command Window** before running your Simulink model

7

Part C) Open Simulink Model (0 Points)

The following model is provided, called *equalpowerpanning_Simulink.slx.* Make sure the block properties are set to the right values and functions, shown below:

Instead of a microphone input, we will be using a "from multimedia file" block to read in violin_44100.wav. **Point this block to where you've saved that audio file**



"to workspace" block with variable name output_equalpowerpanning

MATLAB FCN pointing to equalpowerpanning_process function

Knob controlling lambda ranges from 0 to 1

MATLAB FCN pointing to lambda_update function

8

Part D) Lambda update function

(0.5 Points)

Complete the code provided in *lambda_update.m* whereby the global variable saving the simulations lambda value is updated when the knob controlling lambda is updated in Simulink.

```
function [] = lambda_update(new_lambda)

% pass by reference:
global equalpowerpanning_lambda;

if equalpowerpanning_lambda ~= new_lambda
    equalpowerpanning_lambda = % FILL IN
end


end
```

9

Part E) Interupt/Process function

(0.5 Points)

Complete the code provided in *equalpowerpanning_process.m* whereby an incoming **mono** audio frame is **split into two channels**, and each channel is scaled by the appropriate coefficient based on the current lambda value. Recall, the coefficients are calculated as follows:

$$y[left] = x\cos\left(\lambda\frac{\pi}{2}\right), \qquad y[right] = x\sin\left(\lambda\frac{\pi}{2}\right),$$

```
function [out_frame] = equalpowerpanning_process(in_frame)

% pass by reference
global equalpowerpanning_lambda;

left_coeff =  % FILL IN   % hint, use cosine here
right_coeff = % FILL IN

% Package the output frame
out_frame = [ (left_coeff * in_frame) (right_coeff * in_frame)];


end
```

10

Part F) Equal Power Panning Demo (1.5 Points)

Run your Simulink model and demonstrate its ability to pan the violin audio left and right using the lambda control knob. Because there is nothing to 'dictate' here, just pan slowly across the range of lambda. We will verify that your result sounds like it's being panned with a simple equal power panning algorithm.

Stop your Simulink model. A script has been provided for you called *demo_audio_equalpowerpanning.m* to convert the audio that was just created in your Simulink model to an audio file, called *equalpowerpanning_demo.wav*. **Run this script in the MATLAB command window after stopping your Simulink simulation**. Make sure you submit this audio file in your submission. EXPLAIN WHAT YOU HEAR IN YOUR REPORT

See the example demo video: https://youtu.be/rj6sgMC2I-g

11

Problem 1 Deliverables

- A graph in your report showing the equal power panning coefficient curves on one set of axes (0.5 points)
- Correctly Implemented lambda_update (0.5 points)
- Correctly Implemented equalpowerpanning_process (0.5 points)
- Submitted demonstration audio that shows a working panner with full range of lambda values (1.5 points)

12

Problem 2: Interaural Time Difference (ITD) and Interaural Intensity Difference (IID)

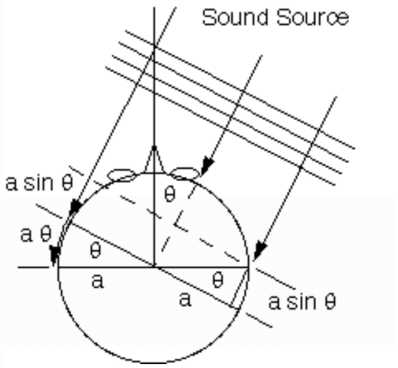
We will be making an ITD-IID panner. This panner should should like the audio is being panned outside of you head

Video Tutorial: https://youtu.be/4jAcU7-YioE

13

Interaural Time Difference

ITD involves delaying the output differently in each ear, so that there is a simulated time difference to when the audio reaches each ear. This is consistent with how audio from real sources reaches the ear at different times based on the path length from source to ear. See the following diagram:



The θ parameter controls the angle the sound source is at relative to the normal (perpendicular) to the axis created by the line between the listener's ears, also called ***azimuth***.

The radius of the human's head is called 'a'. Recall that the speed of sound in air 'c' is roughly 343 m/s

How to use theta to determine the relative delay between the two ears is shown on the next slide

14

Interaural Time Difference

Calculation of the interaural time difference **IN SECONDS** is given by the following formula:

$$\mathrm{ITD} = \frac{a}{c}(\theta + \sin\theta),$$

Theta, called the azimuth angle, ranges in the range $[\frac{-\pi}{2}, \frac{+\pi}{2}]$ in radians. Using the radius of the human's head ('a'), the speed of sound ('c'), and the azimuth angle to determine how long in seconds the delay between the two ears should be. Multiplying this value by fs will give the delay in samples between the two ears. This operation will be done in the *theta_update.m* script, which will be written as part of this problem.

15

Interaural Intensity Difference

Similar to the equal power panning algorithm, in addition to ITD, a level or intensity difference will be used to send different power levels to each ear. The following equations show how this difference is calculated, using the azimuth angle

Coefficient of right sample: $1 + \frac{1}{3}\sin\theta$

Coefficient of left sample: $1 - \frac{1}{3}\sin\theta$

16

Part A) Updating DelayLine.m

(0.5 Points)

An updated version of *DelayLine.m* has been provided for you. A third property has been added, called ITD_delay, which will save how many samples past the currentPos value in the buffer the ITD delayed sample should be read. The process function also now outputs two values, xDelayed and xMoreDelayed. One ear will get the regular output of the DelayLine, the other ear will get a more delayed sample.

The new DelayLine.m has been finished for you however you need to understand and explain what it's doing. In your report, answer the following two questions:

1. What do lines 36-41 (in the process function) do in the provided code
2. Calculate the maximum number in both **seconds and samples** that will be delayed between the two ears based on the azimuth angle's range.
3. Why is the absolute value of ITD_delay taken in these lines?

17

Part B) Initialization

(0 Points)

Look at the provided initialization script *ITD_IID_init.m*. This scripts defines several global variables that will be passed by reference into the Simulink model. A global variable for theta is defined and initialized to zero (center). The radius of the human head is set to 0.1 meters, which is roughly the worldwide average for this value. A long DelayLine (500 samples) is used. You do not need to write anything in this script, but you do **need to run it in the MATLAB Command Window** before running your Simulink Model

```
global ITD_IID_theta;
global a;
global c;
global ITD_DelayLine;
global fs;

ITD_IID_theta = 0;
a = 0.1;
c = 343;
ITD_DelayLine = DelayLine(500);
fs = 44100;
```

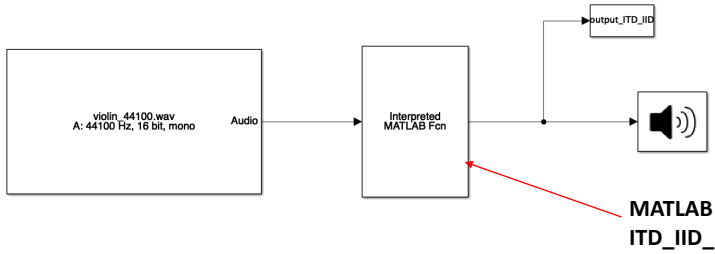
18

Part C) Open Simulink Model

(0 Points)

The following model is provided, called *ITD_IID_Simulink.slx.* Make sure the block properties are set to the right values and functions, shown below:

Instead of a microphone input, we will be using a "from multimedia file" block to read in violin_44100.wav. **Point this block to where you've saved that audio file**

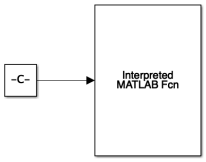


"to workspace" block with variable name output_ITD_IID

MATLAB FCN pointing to ITD_IID_process function



Knob controlling theta ranges from $[\frac{-\pi}{2}, \frac{+\pi}{2}]$



MATLAB FCN pointing to theta_update function

19

Part D) Theta update function

(0.5 Points)

Fill in the blanks in *theta_update.m*. In line 12, you are asked to compute the value in seconds for the interaural time difference between the two ears. At this point, it does not matter which ear is more delayed, remember that your DelayLine uses the absolute value of its ITD_delay property. **Use the previously given formula to compute the delay in seconds between the two ears.** The global variable containing your DelayLine then writes over its ITD_delay property with the new value in line 14

20

Part E) Interrupt/Process function (0.5 Points)

Fill in the blanks in *ITD_IID_process.m.* In lines 9 and 10, use the given IID coefficient equation to set the level difference between the two ears. Then, in lines 20-30, process the audio through the delay line. Notice how the samples labeled left_sample and right_sample change if they are xDelayed or xMoreDelayed based on the value of theta. Appropriately call the process function in these lines.

Reminder: If a MATLAB function has two outputs, but is called so that only one output is written to [ie, >> x = myfunction(…) instead of >> [x y] = myfunction(…)], only the first output argument of that function is saved. We take advantage of this for samples in the center, when theta = 0.

In your report, explain why the order of left_sample and right_sample changes with the sign of theta.

21

Part F) ITD/IID Panning Demo (1.5 Points)

Run your Simulink model and demonstrate its ability to pan the violin audio left and right using the theta control knob. **It should sound like the audio is being panned outside of your head.** Because there is nothing to 'dictate' here, just pan slowly across the range of theta. We will verify that your result sounds like ITD/IID panning.

Stop your Simulink model. A script has been provided for you called *demo_audio_ITD_IID.m* to convert the audio that was just created in your Simulink model to an audio file, called *ITD_IID_demo.wav*. **Run this script in the MATLAB command window after stopping your Simulink simulation**. Make sure you submit this audio file in your submission. EXPLAIN WHAT YOU HEAR IN YOUR REPORT

See the example demo video: https://youtu.be/rj6sgMC2I-g

22

Problem 2 Deliverables

- Part A: Answers about questions regarding the re-writing of DelayLine.m to output two delayed samples (0.5 points)
- Correctly Implemented theta_update, and answer to the maximum delay in seconds and samples of ITD_Delay (0.5 points)
- Correctly Implemented ITD_IID_process, and answer to question in Part E about the order of left_sample and right_sample when calling the HRTFPlayer's process function (0.5 points)
- Submitted demonstration audio that shows a working panner with full range of theta values (1.5 points)

23

Problem 3: Head Related Transfer Functions (HRTFs)

We will be making a panner using given Head Related Impulse Response (HRIR) data. This panner should should even more realistic than ITD/IID panning

Video Tutorial: https://youtu.be/KdZ3VhvR_g4

24

HRTF/HRIR Data

In spatial audio rendering, it is often ideal to think of sources in terms of there azimuth angle (as described the previous problem) and elevation angle, corresponding to the sources' height above or below the listener's position. To measure an HRIR, an impulse is played by a speaker in the physical location described by each combination of azimuth and elevation angle. The speaker should be equidistant to the listener at all positions, meaning we are moving the speaker along a **sphere**.

You are given a '.mat' file called IRC_1002_R_HRIR.mat, which contains experimental Head Related Impulse Responses (HRIRs) for several values of azimuth and elevation angle. We will be exploring these in Part A of this question in the next slide.

25

Part A) Loading HRIR Data (0 Points)

In the MATLAB Command Window, use the load function to load into the workspace the two structs called r_hrir_S and l_hrir_S that contain the right and left HRIR data, respectively

```
>> load('IRC_1002_R_HRIR.mat')
```

You will now have r_hrir_S and l_hrir_S in your workspace. See the next slide

26

Part A) Loading HRIR Data *(Continued)*

```
>> r_hrir_S

r_hrir_S =

  struct with fields:

        type_s: 'FIR'
        elev_v: [187×1 double]
        azim_v: [187×1 double]
   sampling_hz: 44100
     content_m: [187×8192 double]
```

```
>> l_hrir_S

l_hrir_S =

  struct with fields:

        type_s: 'FIR'
        elev_v: [187×1 double]
        azim_v: [187×1 double]
   sampling_hz: 44100
     content_m: [187×8192 double]
```

The content_m arrays contain the 8192-sample long HRIR for the consident elevation and azimuth angle, ie, r_hrir_S.content_m(INDEX,:) contains the HRIR from elevation r_hrir_S.elev_v(INDEX) and azimuth r_hrir_S.azim_v(INDEX). Both struct's elev_v and azim_v arrays are identical, meaning all data is saved in the same order in both structs.

27

Part B) Elevation and Azimuth (0.5 Points)

In your report, use the provided structs detail which elevation and azimuth angle values you have impulse responses for.

28

Part C) Plotting HRIRs (0.5 Points)

In your report, show a subplot of dimensions (1,2,x) whereby the plot shows side by side the left HRIR and right HRIR of three different (azimuth, elevation) combinations. The plots should be 8192 samples long. Make the following observations in your report:

1. For each plot, based on azimuth angle, are the relative levels of the HRIRs in the left and right ear consistent with what you expect?
2. It should look like there are 240 samples between the beginning of the HRIR and when useful data is recorded. If the sampling frequency of the experiment is 44100 Hz, how far away is the source in meters?
3. What happens to the impulse responses after around 500 samples? Is this expected (hint, this is an *impulse* response)

29

Part D) extract_HRIR_data (0.5 Points)

The given script *extract_HRIR_data.m* has been provided to define several global variables that will be passed by reference to the Simulink model. Part of this involves picking out relevant sections of data. Although this script is provided for you, please answer the following questions **in your report**:

1. What do lines 22 and 23 of this script do? What elevation will all the samples be at after running this script?
2. What do lines 26 and 27 do? How does this increase the efficiency of this implementation?

30

Part E) HRTFPlayer (0.5 Points)

We will use these impulse responses to determine what the output in each ear should be by convolving the HRIR of each ear with the input audio. Because the audio comes in in samples, we merely put a scaled version of HRIR into a buffer starting at the currentPos of the buffer **(convolution of a scaled impulse with a response is just a scaled version of the response).** This is similar to a DelayLine, however we are now writing to samples past the currentPos in the buffer, and we now write before we read. Although this class is provided for you, please answer the following questions in the following slide **in your report**:

31

Part E) HRTFPlayer *(Continued)*

Describe the overall algorithm that is being presented in the process function. Break it up into steps and describe each step in detail. These steps should answer things like:

1. Why is the input to the function an array?
2. What does the if statement from lines 30-40 do?
3. How are things being written to the buffer?
4. What is being played at any given point in time?
5. Does the input to the process function (the scaled HRIR) begin to contribute to the output immediately or does it take one full buffer of latency?
6. Why do we set the buffer at the current position to 0 after playing?

32

Part F) Initialization

(0 Points)

Look at the provided script *HRTF_init.m.* In this script, several global variables get defined to be passed by reference into the Simulink model. The script from Part D, *extract_HRIR_data*, is called as a part of this script.

The global variable azim_idx will be updated via a knob in the Simulink model. This variable will be used to determine which index in the global azimuth_values array we are looking at, such that azimuth_values(azim_idx) is the azimuth angle in degrees. We can also use azim_idx to index the content_r and content_l arrays, which are matrices that save the HRIRs of each azimuth (recalling that elevation is fixed to 0 degrees).

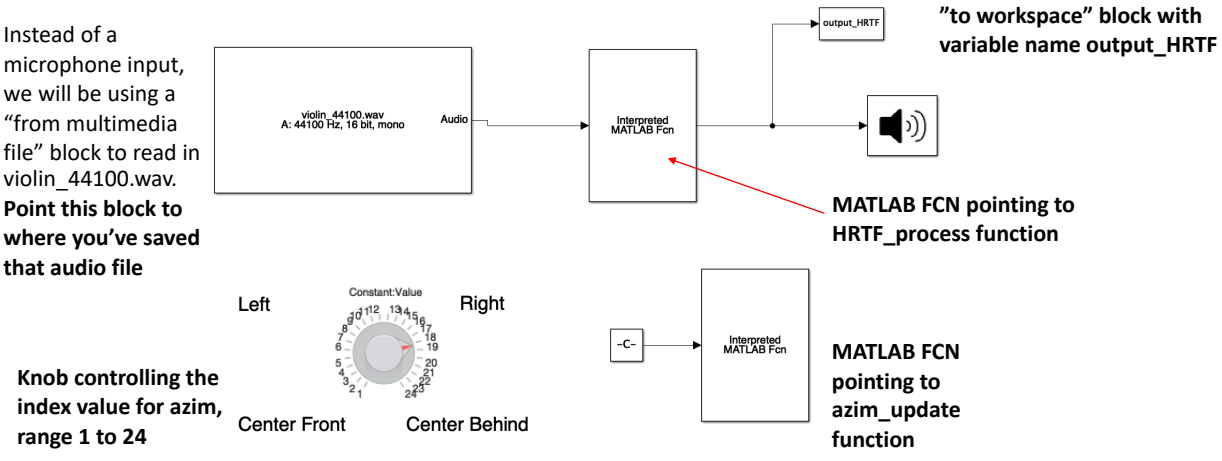
You do not need to write anything in this script, but you do **need to run it in the MATLAB Command Window** before running your Simulink Model

33

Part G) Open Simulink Model

(0 Points)

The following model is provided, called *HRTF_Simulink.slx.* Make sure the block properties are set to the right values and functions, shown below:



34

Part H) Azim Update Function (0.5 Points)

Fill in the blanks in *azim_update.m*. Very simply, every time the knob value changes, azim_idx should change. Since azim_idx is an index, it needs to be an integer, so force this to be the case, bearing in mind the knob might not always give integer values to the FCN block.

35

Part I) Interrupt/Process function (0.5 Points)

Fill in the blanks in *HRTF_process.m.* In lines 18 and 19, you have to prepare two arrays (right_array and left_array) to be processed by yoru global HRTFPlayer objects. To do this, you need to multiply the input sample value (which is a scaled impulse as mentioned previously) with the HRIR (stored in r_content and l_content, accessibly by properly indexing these arrays using the azim_idx).

36

Part J) HRTFPanning Demo (1.5 Points)

Run your Simulink model and demonstrate its ability to pan the violin audio left and right using the theta control knob. **It should sound like the audio is being panned outside of your head.** Because there is nothing to 'dictate' here, just pan slowly across the range of azim_idx. We will verify that your result sounds like HRTF panning. The knob is labled, you should be able to hear panning not just to the right and left, but forward and behind as well. You will be more sensitive to the right and left variation, however.

Stop your Simulink model. A script has been provided for you called *demo_audio_HRTF.m* to convert the audio that was just created in your Simulink model to an audio file, called *HRTF_demo.wav*. **Run this script in the MATLAB command window after stopping your Simulink simulation**. Make sure you submit this audio file in your submission. EXPLAIN WHAT YOU HEAR IN YOUR REPORT

See the example demo video: https://youtu.be/rj6sgMC2I-g

37

Problem 3 Deliverables

- Answer Part B questions: which azimuth and elevation angles do we have HRIR data from (0.5 points)
- Provide subplots of three left and right HRIRs and answer all questions from Part C (0.5 points)
- Answer Part D questions about the operation of the provided script extract_HRIR_data (0.5 points)
- Answer Part E questions about the operation of the provided class HRTFPlayer (0.5 points)
- Correctly Implemented azim_update (0.5 points)
- Correctly Implemented HRTF_process (0.5 points)
- Submitted demonstration audio that shows a working panner with full range of azim_idx values (1 point)

38