TP3 - Procesamiento de Señales

**1. Exportar un archivo wav desde Matlab**

a. Importe desde Matlab el archivo de audio denominado 'unodostres.wav' en una variable denominada unodostres.



b. Invierta temporalmente el audio para que se escuche "hacia atrás" y coloque el resultado en la variable unodostres.

c. Exporte el resultado como una archivo wav denominado "audioInvertido.wav" utilizando la función audiowrite. (Utilice el help de la línea de comandos: >>help audiowrite)

d. Abra y escuche el archivo desde una aplicación diferente (Goldwave, por ejemplo) para verificar que se logró el resultado esperado.

**2. Niveles de saturación**

a. Copie la variable unodostres en otra variable denominada audio.

b. Genere una variable denominada audioSaturado que sea igual a audio multiplicada por 10.

c. Genere una nueva variable que concatene audio y luego audioSaturado en una sola variable de nombre audioTotal.

d. Grafique la variable audioTotal.

e. Tenga cuidado con esta parte si está escuchando con auriculares ya que hemos amplificado mucho parte del sonido. En caso de usar auriculares sugerimos alejarlos de los oídos un poco el primer intento de esta prueba. Si utiliza altavoces no se trata de un nivel problemático. Reproduzca audioTotal en MATLAB. 1) ¿Se escucha correctamente?, 2) ¿Se nota la diferencia de nivel?, 3) Observe en el gráfico que valor numérico alcanza la señal en su punto más alto.

f. Exporte el archivo audioTotal en formato wav.

g. Abra el archivo audioTotal.wav una aplicación diferente que le permita observar la forma de onda (Goldwave, por ejemplo). ¿Qué se escucha y qué se observa? ¿Por qué?

h. Importe el archivo audioTotal.wav en Matlab dentro de una variable denominada audioTotal2.

i. Grafique en dos figuras separadas las variables audioTotal y audioTotal2 (no es necesario agregar el eje temporal), pero si se pide que los ejes para ambas figuras muestren valores entre -3 y 3 (con la instrucción axis)

j. Escriba un breve resumen como conclusión de lo analizado. ¿Cuál es el valor que corresponde al nivel de saturación de un audio? ¿Qué hace Matlab cuando se pide escuchar un archivo de audio cuyo nivel la saturación? ¿Qué hace Matlab cuando exporta un archivo de audio que supera el nivel de saturación?

**3. Uso de la instrucción soundsc**

a. Utilice la variable audioTotal para poner a prueba la diferencia entre la instrucción sound() y la instrucción soundsc(). La idea es que pruebe de escuchar el resultado de utilizar sound() en forma directa desde la línea de comandos, espere un momento hasta que se termine el audio e inmediatamente repita el proceso con soundsc(). No olvide incluir la variable fs en ambos casos. ¿Qué diferencia se escucha? ¿Cuál podría ser el motivo?

b. Genere una nueva variable audioExtendido que combine los sonidos de audioTotal y audioTotal2. Se pretende un archivo monoaural del doble de duración. Utilice nuevamente soundsc() para escuchar la variable audioExtendido. ¿Qué diferencias se presentan con lo que sucedió cuando se ejecutaron por separado cada uno de los audios con soundsc? Explique cuál puede ser el motivo.

c. Genere una variable audioBajoNivel que sea igual a unodostres dividido por 100. Escuche en primer lugar audioBajoNivel utilizando sound y luego utilizando soundsc. ¿A qué se debe la diferencia en lo que se escucha?

**4. Audio estéreo**

a. Utilice clear y luego clc. Importe el archivo "fragmentoQuadrophenia.wav" y guárdelo en la variable "audio".

b. Reproduzca la variable audio en MATLAB.

c. Genere un eje de tiempos que abarque toda la longitud del archivo.

d. Grafique la variable con eje temporal.

e. ¿Qué significan las diferencias de color en algunas partes del gráfico?

*NOTA: Un archivo de audio estéreo importado en Matlab genera una matriz formada por dos vectores verticales (n x 2). El primero corresponde al canal izquierdo y el segundo al canal derecho.*

*Para poder operar cómodamente con este tipo de datos es necesario retomar y profundizar algunas cuestiones trabajadas en los primeros ejercicios.*

*La siguiente instrucción genera una matriz de 3x2 (3 filas y 2 columnas)*

>> a=[1 2; 3 4; 5 6]

a =

1 2

3 4

5 6

*Si necesitamos extraer un elemento de esta matriz, podemos hacerlo especificando el número de fila y de columna. Así, por ejemplo a(1,1) será el número 1, mientras que a(2,1) será el número 3 y a(1,2) será el número 2. (Mostramos como ejemplo el resultado de seleccionar a(2,1) que significa el elemento de fila 2 y la columna 1.*

>> a(2,1)

ans =

3

*¿Cuál será el elemento de la fila 3 y la columna 2?*

*Es posible también seleccionar varios elementos como en el siguiente caso*

>> a( 2:3 , 1 )

ans =

3

5

*Aquí se han solicitado los elementos de la fila 2 y 3, columna 1.*

*¿Qué se obtendría solicitando a(1:2,2)?*

*¿Qué se obtendría solicitando a(2,1:2)?*

*¿Cómo se indicaría toda la columna 1? ¿Cómo se indicaría toda la columna 2?*

*¿Cómo se indicaría toda la fila 3?*

*Matlab permite utilizar la palabra reservada "end" para indicar el último elemento de una fila o columna. De modo que en el ejemplo anterior sería equivalente escribir a(2:end,1) y a(2:3,1).*

*Además cuando se necesita toda una fila o toda una columna es posible colocar allí solamente ":". Para obtener toda la columna 2 serían equivalentes estas tres instrucciones*

>> a(1:3,2)

>> a(1:end,2)

>> a(:,2)

**5. Separación de canales e inversión de canales**

a. Utilice la variable "audio" del ejercicio anterior y obtenga una variable llamada izquierda que contenga la primera columna y una variable llamada derecha que contenga la segunda columna.

b. Utilice la función sound para escuchar la variable izquierda. ¿Se escucha solamente en el canal izquierdo? ¿Por qué?

c. Genere una nueva variable de nombre audioIzquierda de 2 columnas que contenga la variable izquierda como la primera columna y complete con ceros la otra columna. Utilice la función sound para escuchar la variable audioIzquierda. ¿Se escucha solamente en el canal izquierdo? ¿Por qué?

d. Genere una nueva variable de nombre audioDerecha de 2 columnas que contenga la variable izquierda, pero esta vez en la segunda columna y complete con ceros la otra columna. Utilice la función sound para escuchar la variable audioDerecha. ¿Se escucha solamente en el canal derecho? ¿Se escucha el izquierdo? ¿Por qué?

e. Genere una nueva variable de nombre audioX de 2 columnas que contenga la variable audioDerecho como la primera columna y audioIzquierdo como la segunda. Escuche el resultado utilizando la función sound. ¿Qué diferencias cree que existen entre el sonido de audioX y el original de audio?

6**. Generar audio estéreo con dos archivos monoaurales**

a. Importe el archivo "unodostres.wav", y genere la variable "uno" con la primera palabra cortando entre 0 y 1 segundos) y la variable "dos" conteniendo la segunda palabra cortando entre 1 y 2 segundos.

b. Verifique en el workspace que "uno" y "dos" tengan la misma longitud. Si no es así revise el paso anterior y corrija los límites de cada una de las partes.

c. Genere un archivo estéreo denominado "unodos" en el que la palabra "uno" sea la primera columna y la palabra "dos" la segunda. Debería obtener una matriz de nx2 elementos.

d. Reproduzca la variable "unodos" usando sound() utilizando preferentemente auriculares. Describa lo que escucha.

e. Exporte el archivo como unodos.wav, ábralo con una aplicación diferente (Goldwave, p. ej.) para verficar que todo el proceso incluida la exportación es correcta.

**7. Generar audio estéreo con dos archivos monoaurales (2)**

a. Importe los archivos "coro1\_bass.wav" y "coro2\_lead.wav".

b. Genere una variable coroEstereo en el cual "coro1" se encuentre a la izquierda y "coro2" a la derecha.

c. Escuche con auriculares utilizando la instrucción sound

d. Intercambie el orden de las columnas en un archivo coroEstereo2 y vuelva a escuchar. Describa las diferencias entre escuchar coroEstereo y coroEstereo2.

**8. Paneo estéreo (por nivel)**

a. Limpie la memoria y la pantalla (clear y clc). Importe los archivos "coro1\_bass.wav" y "coro2\_lead.wav" en dos variables de nombres pista1 y pista2.

b. genere una variable track1paneoIzq de 2 columnas. Coloque en la primera columna la variable track1 y en la segunda esa misma variable multiplicada por 0,2.

c. genere una variable track2paneoDer de 2 columnas. Coloque en la primera columna la variable track2 multiplicada por 0,2 y en la segunda la variable track2 sin modificar.

d. Escuche la variable track1paneoIzq utilizando sound, preferentemente con auriculares. Luego repita lo mismo con la variable track2paneoDer.

d. Genere una nueva variable audioEstereo que sea igual a la suma de ambas variables. Escuche el resultado.

*NOTA: El proceso realizado se corresponde con un paneo por nivel en el cual la aparente ubicación hacia la izquierda o derecha de una fuente sonora está dada por la ganancia con la que la información de dicha fuente se copia en uno u otro canal del archivo estéreo.*

*NOTA:* ***Scripts y funciones en MATLAB.***

*Hemos visto que para realizar una mezcla estéreo es necesario en principio generar copias de cada canal de entrada (variable de audio) en dos vectores (izquierdo y derecho) con diferente nivel y luego sumar los archivos para obtener la mezcla final.*

*Recordemos primero el uso de una secuencia de comandos (script) con un ejemplo que tome la variable "coro1" y genere coro1Izquierda y coro1derecha, utilizando las variables nivelIzquierdo y nivelDerecho.*

%Ejemplo de script

nivelIzquierda = 0.875;

nivelDerecha = 0.125;

coro1Izquierda = nivelIzquierda\*coro1;

coro1Derecha = nivelDerecha\*coro1;

coro1Estereo = [coro1Izquierda coro1Derecha];

*La primera línea del script comienza con %, esto hace que MATLAB ignore lo que sigue, sólo sirve como "comentario" útil para el programador cuando analice este código más adelante.*

*Las variables nivelIzquierda y nivelDerecha contienen un valor entre 0 y 1 que indica cuánto de la variable coro1 se enviará a cada canal de la señal estéreo.*

*Notar que coro1Estereo se forma por dos vectores columna, por este motivo la concatenación de ambos vectores columnas no debe tener un ";" en medio entre ellos. Si se concatenaran con ";" lo que se lograría es una variable conteniendo ambos audios uno después del otro.*

*Si se guarda este script con el nombre "ejemploScript.m", para ejecutarlo se escribe simplemente en la línea de comando el nombre*

>> ejemploScript

*Si luego se ejecuta plot(coro1Estereo) deberá verse lo siguiente*



*Veamos ahora estas mismas instrucciones formando para de una "función" de Matlab.*

function [ pistaEstereo ] = paneoEstereo( audioMono, nivelIzquierda, nivelDerecha )

%Ejemplo de función en MATLAB

audioIzquierda = nivelIzquierda\*audioMoo;

audioDerecha = nivelDerecha\*audioMono;

pistaEstereo = [audioIzquierda audioDerecha];

end

*El nombre de la función es "paneoEstereo". La primera línea indica que paneoEstereo recibirá la información correspondiente a tres variables: audioMono, nivelIzquierda y nivelDerecha. Operará con estas variables y devolverá un archivo denominado "audioEstereo".*

*Una de las ventajas de una función es que los nombres de las variables se refieren a cómo se llaman dentro de la función, pero pueden cambiarse al ejecutar la función desde la línea de comandos. Esto permitirá usar la misma función para coro1 y para coro2, y también permitirá modificar los niveles de paneo sin tener que volver a escribir la función.*

*Para que se entienda esto mostraremos cómo se ejecuta esa función para realizar la misma tarea que el script del ejemplo anterior.*

>> coro1Estereo = paneoEstereo(coro1, 0.125, 0.875);

*Esto debe interpretarse del siguiente modo. Se está solicitando ejecución de la función paneoEstereo, esta función tiene tres variables de entrada. La primera en el llamado se denomina "coro1", mientras que la primera en la definición se denomina "audioMono". Esto hará que los datos de "coro1" se copien automáticamente dentro de la función en una nueva variable (llamada variable local) que se llama "audioMono".*

*De modo semejante la segunda variable en la definición es nivelIzquierda, mientras que el segundo dato en la llamada es un número. Esto provocará que este número se copie en la variable local "nivelIzquierdo". Lo mismo sucede con el tercer valor de argumentos de entrada.*

*La función se ejecuta luego con estas variables locales y el resultado se guarda en la variable local paneoEstereo. Esta variable local es la "variable de salida" de la función. Es la información que devuelve luego de ejecutarse. Esta variable se denomina "paneoEstereo" dentro de la función y tiene el nombre "coro1Estereo" en la instrucción de la línea de comandos. En este caso lo que sucederá es que la información de paneoEstereo se guarda en la variable coro1Estereo.*

*Dicho así, suena complicado, pero es relativamente sencillo captar la idea luego de algunas pruebas.*

*La función se guarda en principio como un archivo aparte (con extensión .m) y debe estar presente en el directorio de trabajo de Matlab. Más adelante veremos cómo hacer que una función pueda utilizarse desde cualquier directorio. Por el momento, la función puede ejecutarse desde la línea de comandos o también desde otro script que se encuentre en el mismo directorio de Matlab.*

**9. Generación de una función propia de paneo por nivel**

a. Genere la función mostrada en el ejemplo del recuadro anterior y guárdela con el nombre "paneoEstereo.m"

b. Ejecute la línea de instrucción recién mencionada

>> coro1Estereo = paneoEstereo(coro1, 0.125, 0.875);

c. Reproduzca el audio y grafique la variable coro1Estereo.

d. Genere una nueva instrucción para generar la variable coro2Estereo que panee coro 1 con valores de 0.7 a la izquierda y 0.3 a la derecha.

e. Obtenga una variable coroEstéreo que resulte de sumar (mezclar) ambas variables estéreo.

f. Reproduzca esta variable y verifique que el paneo estéreo se haya logrado a nivel perceptivo.

NOTA: El paneo por nivel que usualmente está incorporado en los distintos sistemas de mezcla está adaptado a una situación en la cual los canales izquierdo y derecho se emiten al aire. En esta situación hay que comprender un problema de variación de nivel al ir paneando el audio de un canal de izquierda a derecha. Cuando el canal está completamente a la izquierda podríamos considerar que tenemos una variable audioEstereo = [1\*audioMono 0\*audioMono]. Al ubicarlo al centro podríamos pensar en generar audioEstereo = [1\*audioMono 1\*audioMono]. El problema de nivel se presenta porque en este caso tanto el parlante izquierdo como el derecho estarán emitiendo la misma señal y por lo tanto se escuchará con más nivel que cuando estaba paneado a la izquierda. Para corregir este problema habría que disminuir el nivel de izquierda y derecha cuando la fuente sonora pretende ubicarse en el centro. Pero, ¿cuánto disminuirlas? ¿Será razonable generar audioEstereo = [0.5\*audioMono 0.5\*audioMono]?

En realidad lo correcto sería compensar el crecimiento de nivel aparente que se produce cuando el audio se emite desde ambas fuentes. Al sumar dos sonidos al aire de igual nivel el nivel total se incrementa 3 dB, por lo cual par que el nivel aparente parezca igual que cuando está a la izquierda, al colocar la fuente en el centro cada canal debería tener 3dB menos. Esto corresponde a utilizar un factor de 0,7071. ¿Por qué? Porque 20.log(0,7071) es igual a -3 dB. De modo que para colocar la fuente al centro lo razonable es usar

>> audioEstereo(0.7071\*audioMono 0.7071\*audioMono];

Ahora bien, ¿qué debería hacerse cuando la fuente está cerca del centro pero tendiendo a uno de los lados? En principio sería necesario asegurar que la suma con la ley de suma de sonidos al aire mantenga la relación de modo que el nivel sea aparentemente igual al de toda la señal a la izquierda.

La ley más común para generar esto se denomina en inglés "sine law panning" o "3dB equal power sine law". Se utiliza una variable x que varía entre 0 (izquierda) y 1 (derecha), y se aplican las siguientes ecuaciones



El siguiente gráfico muestra los niveles Izquierdo y Derecho para cada valor de x de modo que el nivel sonoro general se mantenga constante.



**10. Función de paneo por nivel mejorada**

a. Genere una función paneoEstereo2 que reciba como variables pistaMono y el valor de x, y entregue el archivo pistaEstereo siguiendo la ley explicada en la nota anterior de igual nivel de sonoridad (sine law panning).

b. Ponga a prueba la función ejecutándola para panear coro1 con las siguientes situaciones: x=0; x=0.25; x=0.5; x=1. Donde x=0 indica completamente paneada hacia un lado y x=1 completamente hacia el otro. En cada caso reproduzca el audio con auriculares para verificar que el paneo es el deseado.

c. Utilice la función recién creada para generar una mezcla entre coro1 con x =0.3 y coro2 con x=0.7. Reproduzca la mezcla generada.

**11. Mezcla de 4 canales**

a. Genere una función llamada mezcla4canales que reciba los cuatro audios monoaurales del coro, los cuatro valores de x (izq-der), y entregue un archivo estéreo con esa mezcla.

Ayuda: La función será del tipo

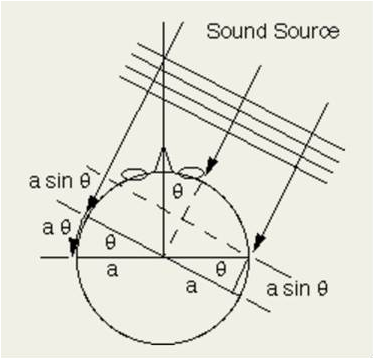
audioMezcla4 = mezcla4canales(pista1,x1,pista2,x2,pista3,3);

b. Utilice la función para hacer una mezcla de los cuatro archivos del coro y reproduzca el resultado.

*NOTA: Las mezclas de audio realizadas en consolas utilizan paneo por nivel. Al escuchar la mezcla con auriculares la posición aparente de cada fuente sonora está determinada por el nivel relativo de intensidad que posee esa fuente en cada canal. Sin embargo eso no se corresponde con lo que sucede cuando existe una fuente real ubicada en cierta posición del espacio.*

*Se sabe que el nivel de presión sonora disminuye 6 dB cada vez que se duplica la distancia. Para que una fuente sonora produjese una caída de 6dB por esta causa entre la señal que llega a cada oído sería necesario que la fuente se encontrase al doble de distancia de un oído que del otro. Esto casi nunca sucede excepto para fuentes muy cercanas al oyente. Si la caída fuese de 12 dB, la distancia de la fuente a un oído debería ser cuatro veces mayor que al otro.*

*Entre los indicios que permiten al cerebro localizar los sonidos se encuentra el retardo que se produce por la diferencia de recorrido desde la fuente a cada oído.*



*Si se considera una cabeza esférica de radio a, el recorrido adicional para alcanzar el oído izquierdo de la figura será la suma del segmento de recta a.sin(θ) más el segmento de circunferencia a.θ*

*El retardo se calcula como distancia sobre velocidad, por lo cual se obtiene*



*donde c es la velocidad de propagación, a es el radio de la cabeza y θ es el ángulo en radianes contado desde el frente hacia la orientación de la fuente sonora.*

*Con esta ecuación es posible calcular qué retardo hay que aplicar en un canal para que la fuente sonora aparente se encuentre localizada en la dirección indicada por el ángulo θ.*

**12. Cálculo del retardo (diferencia temporal interaural)**

a. Calcule el retardo temporal entre oídos provocado por una fuente ubicada a 45º. Considere que el radio de la cabeza es a= 0.09 m (9 cm). Recuerde que para el cálculo, el ángulo debe ser colocado en radianes en la ecuación de la nota anterior.

b. El máximo retardo posible se producirá cuando la fuente esté a 90º del frente (apuntando directamente a uno de los oídos). Calcule el valor de este retardo.

*NOTA: Conociendo esta ecuación es posible pensar en realizar un paneo por retardo temporal (conocido como ITD, interaural time difference). Para ello se necesitaría copiar la señal sin modificar en uno de los canales del audio estéreo y colocar una copia con un retardo en el otro canal.*

*¿Cómo se agrega un retardo? Generando una variable compuesta que contenga una cantidad de ceros determinada y luego la información de audio.*

*¿Cuántos ceros habrá que agregar? Esto dependerá de la frecuencia de muestreo fs. Si quisiéramos un retardo de 1 segundo, se necesitarían fs muestras. Si se quiere un retardo de 0,1 segundos se requieren 0,1.fs muestras. De modo que si se necesita un retardo Δt, la cantidad de muestras será Δt\*fs, teniendo cuidado de quitar al resultado los decimales, ya que el número de muestras debe ser un entero. De este modo será nmuestras = floor(deltat\*fs);*

*¿Cómo generar una variable que contenga una cantidad deseada de ceros? Existen varias alternativas. Una de ellas es generar un vector de nmuestras y multiplicar todos sus valores por cero. Pero dado que las variables de audio son vectores verticales, hay que asegurarse que el vector de n muestras sea vertical.*

*Ejemplo:*

>> retardo = (1:nmuestras)\*0; % Esta instrucción genera un vector horizontal y multiplica todo por cero.

>> retardo = retardo.' % Esta instrucción realiza la transposición del vector (lo convierte en vertical).

*Otra manera es utilizar una instrucción especial de Matlab denominada zeros(n,m). Esta instrucción genera una matriz de ceros de n columnas y m filas.*

*Ejemplo*

>> retardo = zeros(nmuestras,1) % Esta instrucción genera directamente un vector columna

*Ejemplo de generación de una variable con retardo incorporado:*

*>> audioconretardo = [zeros(nmuestras,1);audio];*

c. Importe el audio "chimesWindows.wav". Grafique la variable sin eje horizontal.

d. Genere una variable chimesconretardo que tenga 0,5 segundos de retardo temporal (ceros al inicio del archivo). Grafique la variable (sin eje horizontal) para verificar que los primeros valores sean ceros.

**13. Localización de audio por ITD (interaural time delay)**

a. Genere una función denominada paneoITDderecha que recibe una variable audioMono, la frecuencia de muestreo y un ángulo en grados y entregue una variable estéreo con un retardo agregado en el canal izquierdo. (Recuerde que antes de aplicar la ecuación de ITD debe calcular el ángulo en radianes).

*NOTA IMPORTANTE: Para poder generar la variable estéreo ambos vectores deben ser iguales. Pero resulta que al haber provocado un retardo en el vector izquierdo esto provocará que sea más largo. Será necesario completar el vector de la derecha con la misma cantidad de ceros al final antes de combinar ambos vectores en una señal estéreo.*

b. Importe el archivo coro1 y genere coro1estereo con paneoITDderecha y un ángulo de 75º.

c. Genere un eje de tiempos que comience en 0, tenga un salto dt=1/fs y recorra todo el tiempo del archivo estéreo generado.

>> longitud = length(coro1estereo(:,1));

>> dt=1/fs;

>> t=0:dt:longitud/fs - dt;

d. Grafique la variable coro1estéreo con su eje de tiempos

d. Utilizando axis([1.85 1.87 -0.5 0.5)) observe un detalle el retardo entre ambas señales



**14. Localización de audio por ITD (2)**

Repita el ejercicio anterior pero creando una función paneoITDizquierda.

NOTA: **Instrucción de control "if" de programación**

En los ejercicios anteriores nos enfrentamos a una situación en la que debemos utilizar una función para el caso de un sonido a la derecha y otra función para un sonido a la izquierda.

¿Por qué resulta complicado utilizar una misma función? ¿Qué tal si considerásemos que el ángulo negativo implica ir hacia la izquierda y positivo ir a la derecha? El problema es que el proceso que realiza la función es el de agregar ceros al principio a uno solo de los canales. ¿Cómo hacer que si el ángulo es negativo agregue ceros a uno de los canales pero si es positivo agregue ceros al otro?

En códigos de programación existe una estructura que se conoce como "if". Se trata de una instrucción que consta de tres partes.

If {condición};

{grupo de instrucciones que se ejecutan si la condición es verdadera}

else

{grupo de instrucciones que se ejecutan si la condición es falsa}

end

La condición es una expresión que puede tener solo dos valores como resultado: verdadero o falso. Un ejemplo de esto es "1 > 0". Esta expresión dice que 1 es mayor que 0 y su resultado es verdadero. Obviamente no tendría sentido utilizar esta condición en una instrucción de tipo IF ya que la condición será siempre verdadera por lo tanto las instrucciones que se ejecuten serán siempre las mismas. Pero si lo que comparamos fuese una variable con un número, o una variable con otra, entonces algunas veces podrían ejecutarse ciertas instrucciones mientras que otras veces se ejecutarían instrucciones diferentes.

Esto puede ser útil para solucionar el problema de la mezcla por ITD, ya que si la variable "angulo" es positiva entonces se agregan ceros en un canal, mientras que si es negativa se agregan ceros en el otro canal.

Ejemplo de instrucciones dentro un script o de una función:

if angulo>0;

audioEstereo = paneoITDderecha(audioMono, fs, angulo);

else

audioEstereo = paneoITDizquierda(audioMono, fs, -angulo);

end

Es importante notar que en la instrucción que está luego del "else" (que se ejecuta "si no es cierto"), la variable angulo tiene un signo menos. Esto se debe a que si no es cierto que angulo es positiva, entonces es negativa y dado que la función paneoITDizquierda fue creada pensando en un valor positivo de ángulo entonces debemos cambiarle el signo antes de ejecutarla.

**15. Localización por ITD (3)**

a. Genere una función denominada paneoITD que reciba la variable audioMono, fs y angulo, pero esta vez considerando que si el ángulo es negativo será porque la fuente está a la izquierda y si es positivo estará a la derecha.

b. Pruebe la función generando un audioEstereo con cualquier audio que desee de los ejemplos ubicando en primer lugar la fuente con angulo = -70º y luego repita con angulo = 70º, para comparar los resultados al reproducir el sonido.

*NOTA: Para poder realizar una mezcla por ITD vamos a tener que sumar las distintas variable paneadas. Pero aquí surge un problema. Al panear por ITD se agregan distintos retardos a los canales y esto produce archivos de longitud diferente. Matlab solo permite sumar vectores o matrices de iguales dimensiones, por lo que tendremos dificultades para sumar dos audios estéreo que tengan longitudes diferentes.*

*Este problema puede solucionarse de varias maneras. Un modo que proponemos es el de generar una función que sume dos audios, teniendo en cuenta que ambos comienzan en forma simultánea y que la duración total será la del archivo de mayor duración.*

*Mostramos a continuación la función sumaAudio() que admite tanto archivos monoaurales como archivos estéreo.*

function [ audio12 ] = sumaAudio( audio1,audio2)

% [ audio12 ] = sumaAudio( audio1,audio2);

% Tomda dos audios (monoaurales o estéreo) y los suma aún cuando tengan

% diferente longitud total.

long1=length(audio1(:,1));

long2=length(audio2(:,1));

if length(audio1(1,:))<2; monoEstereo=1; else monoEstereo=2; end

long=max([long1 long2]);

audio12=zeros(long,monoEstereo);

audio12(1:long1,:)=audio1;

audio12(1:long2,:)=audio12(1:long2,:)+audio2;

end

*La función fue definida para sumar dos variables, en caso de necesitar sumar más variables habrá que volver a aplicar la suma del siguiente modo*

>> audio12=sumaAudio(audio1,audio2);

>> audio123=sumaAudio(audio12,audio3);

**16. Mezcla por ITD**

a. Genere una función denominada mezclaITDx4 que reciba cuatro variables de audioMono y cuatro ángulos correspondientes a cada uno de los audios y devuelva la mezcla por ITD. Necesitará utilizar la función sumaAudio que se menciona en la NOTA anterior (u otra equivalente).

b. Pruebe la función creando una variable coroITD que contenga la mezcla de los cuatro audios de coro ubicando las fuentes a -75º, -30º, 30º y 70º. Reproduzca el audio

**17. Comparación de mezclas ILD e ITD**

a. Tomando como base la función de mezcla del ejercicio 35 (paneoEstereo2.m) genere una nueva función denominada paneoILD que reciba audioMono y el ángulo de inclinación (negativo para izquierda y positivo para derecha). El término ILD hacer referencia a "intensity level difference" (diferencia de nivel de intensidad). El ángulo que indica qué tan inclinado hacia izquierda o derecha se encuentra la fuente se denomina ángulo de acimut (azimuth en inglés), para diferenciarlo del ángulo de elevación (elevation en inglés) que indicaría qué tan arriba o abajo está una fuente sonora.

Para poder resolver este punto es necesario convertir el ángulo de acimut en un valor de x (0 para izquierda y 1 para derecha). Esto puede hacerse mediante la siguiente expresión

>> x = angulo/180 + 0.5;

b. Genere una función mezclaILDx4 que reciba cuatro canales de audioMono y los cuatro ángulos correspondientes y entregue una mezcla por nivel.

c. Genere una variable coroILD que contenga la mezcla estéreo por nivel de los cuatro audios del ejercicio anterior considerando esos mismos ángulos (-75, -30, 30 y 75)

d. Reproduzca ambos audios uno a continuación del otro para comparar los efectos de las mezclas

>> sound( [coroILD; coroITD] , fs)

*NOTA: Al comprar podrá notar que las mezclas por ITD no logran en ningún caso ubicar a las fuentes aparentes muy a la izquierda o muy a la derecha. Esto provocará que las mezclas por ILD generen mayor separación angular aparente. Como discutiremos más adelante, una buena representación de audio binaural requerirá incorporar más elementos que solamente el retardo temporal. Estos serán el efecto de difracción provocado por la cabeza que genera una caída en altas frecuencias sobre el oído contralateral, y el efecto de alteración del espectro producido por el pabellón de la oreja, toda la cabeza y el torso. Estos efectos están incluidos en lo que se conoce como "función de transferencia de la cabeza" o HRTF (head related transfer function).*

**18. Versión simple de karaoke**

Una versión muy simple de karaoke (que pretenda quitar la voz principal manteniendo los instrumentos) puede funcionar en mezclas muy sencillas que tengan la voz principal paneada por ild exactamente al centro, mientras los instrumentos se encuentren localizados con algún ángulo de acimut.

Si este es el caso se tendrá que la información de la voz es exactamente igual en ambos canales, mientras que el resto de la información es mayor en uno de los canales. Sabiendo esto, al restar los canales entre sí se consigue un archivo monoaural en el cual la voz queda suprimida.

a. Importe el archivo "In My Life - mezcla elemental 090319.wav" y cambie el nombre autogenerado "data" por "inmylife". Reprodúzcalo. Como tiene una duración de algunos minutos, ejecute la siguiente instrucción para detener el audio

>> clear sound

b. Separe los canales en una variable izq y otra variable der.

c. Genere una versión monoaural de nombre "mono" sumando estas dos variables. Reprodúzcalo y deténgalo cuando lo desee.

d. Genere una versión sin las voces creando la variable

>> karaoke=izq-der;

e. Reproduzca el audio de karaoke. Notará que las voces prácticamente han desaparecido. ¿Por qué cree que de todos modos se escucha algo de ellas a muy bajo nivel?

f. Genere una función generaKaraoke que reciba audio estéreo y entregue un archivo mono que utilice este procedimiento simple de karaoke.

*NOTA: Convertir un archivo estéreo a mono puede generar problemas cuando existen diferencias de tiempo en las señales entre ambos canales. Al sumar dos señales con diferencias de fase, se incrementarán algunas frecuencias y se cancelarán otras. Esto altera el espectro de la señal ecualizándolo de una manera no deseada. Una grabación estéreo realizada en vivo mediante un par espaciado de micrófonos tendrá diferencias de tiempos entre canales para las señales de cada instrumento, y esto provocará esa ecualización indeseada. Si, en cambio, la imagen estéreo se ha generado mediante una mezcla por niveles (ild), entonces todas las señales están sincronizadas y la suma de ellas no provocará este efecto. En las primeras grabaciones estéreo comerciales se aclaraba esto indicando que se trataba de una grabación "monocompatible", que significaba que era estéreo pero que no quedaría ecualizada al convertirla en monoaural.*

*Es importante tener esto claro cuando se generan mezclas por itd (o mezclas binaurales más complejas mediante HRTF) ya que en ese caso una versión monoaural generada sumando ambos canales tendrá este efecto de ecualización. El resultado es muy bueno con auriculares en estéreo, pierde algo del efecto al ser emitido al aire con altavoces y puede ser muy malo si se convierte la señal a monoaural.*