UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA

INGENIERÍA ELÉCTRICA – PROCESAMIENTO DIGITAL DE SEÑALES

Localización de una fuente de voz con arreglo de micrófonos

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRO EN INGENIERÍA

P R E S E N T A

AGUSTÍN SILVA LANG

DIRECTOR DE TESIS

M. I. Larry Hipólito Escobar Salguero

MÉXICO, D. F. MARZO 2014

JURADO ASIGNADO:

Presidente: **Dr. Mario Peña Cabrera**

Secretario: **Dr. Pablo Roberto Pérez Alcázar**

Vocal: **M. I. Larry Hipólito Escobar Salguero**

1 er. Suplente: **Dr. Carlos Rivera Rivera**

2 do. Suplente: **Dra. Fatima Moumtadi**

Lugar o lugares donde se realizó la tesis:

Laboratorio de procesamiento digital de señales, Posgrado de Ingeniería UNAM.

México DF.

TUTOR DE TESIS:

M. I. Larry Hipólito Escobar Salguero

FIRMA

Agradecimientos

A mis padres José Amado Silva Pedraza y Aida Lang Baez gracias por intentar dar su apoyo en estos últimos años.

A mi hermano Enrique sin tu apoyo y ayuda no lo hubiera logrado.

Al M. I. Larry H. Escobar Salguero muchas gracias por su apoyo, paciencia y darme la oportunidad de realizar esta maestría la cual no habría podido realizar sin su ayuda.

Gracias a los amigos Luis Javier Vargas Bautista, Irma Cecilia Díaz Rojas y Edgar Alonso Trueba por sus ánimos y amistad durante el proceso de este trabajo.

RESUMEN

El objetivo de la tesis fue diseñar un arreglo de tres micrófonos que es capaz de detectar la posición angular de una señal de voz entre una fuente de sonido y el arreglo con el DSP TMS320C6416.

Se investigaron métodos actuales de 2 o tres micrófonos que hacen la detección de ángulo, de los que se mencionan la familia de métodos correlación cruzada generalizada GCC, algoritmo descomposición de eigenvalores adaptables, la familia de métodos por funciones de transferencia relacionadas a la cabeza HRTF, de los cuales se eligió utilizar el método de GCC Sumado y el transformada de fase PHAT, para comparar su desempeño y posible aplicación.

El arreglo de micrófonos consistió de tres micrófonos electret em-926 colocados en un plano distribuidos sobre los vértices de un triángulo equilátero de 17 centímetros de cada lado, cada micrófono recibe el nombre dependiendo de su ubicación siendo derecho R, izquierdo L y atrás B. Los cuales forman tres sub arreglos de dos micrófonos.

Los métodos GCC Sumado y la transformada PHAT cuya base es la correlación cruzada en el tiempo y en la frecuencia respectivamente; con las señales obtenidas del arreglo pueden efectuar una estimación del ángulo donde se localiza la fuente de sonido en teoría de -180° a 180° de una circunferencia sobre el plano donde se ubica el arreglo de micrófonos.

Se elaboró un programa en el Matlab para la prueba de los métodos, en estas pruebas se observó que con señales sintéticas se efectuó la localización, en base a dichas pruebas se procedió a la construcción física del arreglo.

Con una bocina conectada a la computadora la cual reproduce un programa de radio se procedió a la captura de las señales en el DSP TMS320F2812 cuyos datos son pasados al DSP TMS320C6416 para su procesamiento. En teoría el procesamiento de datos con el GCC sumado se puede efectuar en 2.7 ms y con la transformada PHAT en aproximadamente 5.3 ms. Al momento de comparar los resultados de ambos métodos se determinó que el método GCC sumado tiene buenos resultados excepto para el intervalo angular de -105° a -70° y de 75° a 115°, en cambio para la transformada PHAT se determinó que aunque puede obtener un estimado pero muy errático.

ABSTRACT

The goal of the thesis was to design an three microphones array that can detect angular position of a voice signal with the array and the DSP TMS320C6416.

It was investigated actual methods of two or three microphones can do angular detection, some of them are family methods generalized cross correlation GCC, adaptable eigenvalues decomposition algorithm, family methods of head related transfer functions, from of them Summed GCC method and Phase transform PHAT were chosen to be compared and possible implementation.

The microphone array consisted of three electret em-926 microphones placed in a plane distributed on the vertices of a equilateral triangle of 17 centimeters per each side, each microphone are named in relation to its position they are, right R, left L, and back B. And they form three sub arrays of two microphones.

The Summed GCC method and PHAT transform based in cross correlation in time and in frequency respectably, with the signals obtained from the array it could get an angular estimation of the sound source location, in theory, in a range of -180° to 180° in a circumference on the plane where it is the microphone array.

It was made a Matlab program to probe the methods, in these probes I was observed that with synthetic signals it is possible the localization, in base to such probes the next step was to build the physically the array.

With a speaker connected to a computer that was playing a broadcast record, it was recorded the signals with the TMS320F2812 DSP, this data was processed by the TMS320C6416 DSP. In theory the data with summed GCC method could be processed in 2.7 ms and with PHAT transforms in approximately 5.3 ms. At the moment to compare the results of both methods was determined that summed GCC has good results except to the angular range of -105° to -70° and 75° to 115°, in change to PHAT transform was determined that could get an estimate but very erratic.

CONTENIDO

1

INTRODUCCIÓN

La localización de una fuente de onda, ya sea acústica o electromagnética, es un campo de interés en el cual, desde hace mucho tiempo, se han planteado distintos enfoques con tal de darle una solución, lo que ha producido distintos métodos a partir de los cuales es posible realizar la detección aunque cada uno de estos métodos con sus respectivas complejidades y restricciones.

En la vida cotidiana, en nuestro ambiente, percibimos infinidad de sonidos, en un ambiente urbano el sonido de los automóviles, maquinaría, la plática de los transeúntes, etc. El ser humano al igual que otros animales, para poder desplazarse en este medio tiene que ser capaz de identificar la presencia de estas entidades. Para ello hacemos uso de nuestros sentidos; la vista, el olfato, el tacto, en algunos casos el gusto y el oído. Aunque el ser humano hace uso principalmente de la vista, quedando en segundo término los demás sentidos; en personas con la pérdida de la vista sus otros sentidos se ven desarrollados para compensar otras capacidades. Las personas con problemas visuales pueden llegar a desarrollar en gran medida su sentido del oído siendo posible para ellos detectar la dirección de otra persona por el ruido que esta hace al desplazarse, a este proceso se le llama sonar. En la naturaleza esto se presenta con frecuencia entre animales, como el murciélago que es capaz de cazar insectos en total oscuridad.

El ser humano puede realizar la localización de fuentes de sonido, cuando alguien se encuentra en un lugar abierto y en ese instante otra persona llama su atención, dando indicaciones de cómo debe transitar pedir información, la primera persona puede identificar donde se encuentra la otra por el sonido que perciben sus oídos y con la vista hace una localización más completa y así poder efectuar decisiones como voltear y dirigirse a donde se encuentra para entablar una conversación, o seguir su camino.

En la actualidad, con los avances en la ciencia y la tecnología se intenta que un robot sea capaz de realizar muchas de las tareas que el ser humano puede realizar, algunas de las cuales hacen uso de la percepción del sonido. Dentro de estas capacidades está ubicar fuentes de sonido y el reconocimiento de voz. Actualmente se están desarrollando robots que puedan interactuar con el hombre, tal es el caso de robots guías en museos, los cuales se desea que puedan

interaccionar con el publico lo más cercano posible como lo haría una persona. El público podría llamar la atención del robot por medio de la voz considerando la posición de la persona, el robot para una interacción con los usuarios tendría que tener un sistema de localización de fuentes de sonido que serviría de apoyo a su sistema de comunicación visual [6].

Además, puede haber otras aplicaciones tales como cámaras de vigilancia que puedan cambiar de objetivo al percibir un sonido para verificar el origen de éste.

Para que una aplicación realice la ubicación de fuentes de sonido se han propuesto varias metodologías utilizando arreglos de micrófonos, desde uno solo hasta varios colocados de forma lineal o en un plano. Al hacer uso de arreglos de micrófonos, en teoría, entre mayor sea el número sensores se tiene una mejor precisión en cuanto a la localización de la fuente; además, el número de fuentes de sonidos posibles a localizar es igual al número de sensores del arreglo. Sin embargo, hay varias limitantes, tales como las dimensiones tanto de los sensores como de los micrófonos, la complejidad de procesamiento de las señales, el costo relacionado al micrófono y el dispositivo dedicado al procesamiento de las señales. Entre más micrófonos, mayores serán las dimensiones del arreglo, y por tanto más pesado, si se utiliza un arreglo grande y pesado conlleva a un aumento de potencia necesaria para el movimiento, además de que el desplazamiento del mismo se vería limitado por el arreglo, por ello es necesario que esto no sea una restricción.

Esta investigación se centra en realizar la localización de fuentes de sonido mediante un arreglo, cuyas dimensiones tamaño y número de micrófonos sean los mínimos, de tal forma que no sea una limitante en aplicaciones donde esto no se requiere.

En este capítulo se exponen los objetivos de esta tesis, de forma breve algunos métodos disponibles para la detección de una fuente de ondas acústicas en específico, así como una descripción de cada uno de los capítulos de esta tesis.

1.1. OBJETIVOS

Desarrollar un sistema, a partir de un arreglo mínimo de micrófonos (2 ó 3 micrófonos), que sea capaz de detectar la posición angular de una la fuente de sonido respecto a un arreglo de micrófonos utilizando una plataforma DSP.

- Investigar los métodos y técnicas actuales utilizados en la detección espacial de voz.

- Investigar las limitantes que se tienen al utilizar solo dos o tres micrófonos en un arreglo para realizar la ubicación de una fuente de sonido.

1.2. PLANTEAMIENTO DEL PROBLEMA

Al trabajar con ondas sonoras, se presentan ciertos fenómenos que hay que considerar, tales como la refracción, la difracción, la atenuación al desplazarse en un medio, la reverberación de las ondas sonoras; así como las fuentes de interferencia que se pueden presentar. Cuando una onda sonora se propaga dentro de una habitación puede encontrar obstáculos, donde al medir

parte de la energía es absorbida, o transmitida a través del material, y otra parte es reflejada, por lo que la onda reflejada es una imagen de la fuente original [15].

Por otra parte, cuando una fuente empieza a emitir sonido en una habitación, la intensidad del sonido o volumen aumenta hasta un nivel estable en un periodo de tiempo entre 0.15 s y 0.25 s. Cuando la fuente de sonido deja de emitir, el sonido decae gradualmente en un tiempo similar; este fenómeno se le llama reverberación, y al tiempo mencionado se le llama tiempo de reverberación. El tiempo de reverberación tiene efectos importantes en la inteligibilidad de la voz y en la calidad del sonido y depende del tamaño de la habitación en el cual la energía de la onda sonora es absorbida por los muros y objetos en su interior. En el caso de la voz, un tiempo corto de reverberación implica una alta absorción, por lo que la voz será difícil de escuchar desde los puntos más alejados al orador. Por otro lado, un tiempo de reverberación largo implica que el sonido de cada sílaba se sobrepone a las silabas previas perdiendo la inteligibilidad del discurso [15].

El frente de onda, de una onda sonora, tiende a ser esférico, así que al ser emitida se propagada en todas direcciones; debido a esto la potencia de la señal se atenúa conforme se aleja de la fuente, de acuerdo a la ley del inverso del cuadrado, la cual dice que al aumentar la distancia en una unidad, la atenuación se duplica [1].

Para realizar la localización de fuentes de sonido, hay que tomar en cuenta estos fenómenos, ya que siempre están presentes y solo en condiciones controladas es posible atenuar sus efectos. Así, si se desea realizar la localización, idealmente solo debería de efectuarse la localización de la fuente de sonido, pero debido a la reflexión de onda puede detectarse una imagen de la fuente como si fuera la original. En cuanto a la reverberación también puede afectar al proceso en gran medida, ya que si el tiempo de reverberación es largo, aunado a las reflexiones de las ondas, puede dificultar el proceso de localización, al provocar interferencias en las señales de entrada.

El sistema efectuará la localización de las fuentes de onda sonora en una habitación en condiciones reales, es decir, donde se presentan los fenómenos antes mencionados. Uno de los mayores problemas que se pueden presentar es la detección de una imagen de la fuente como si fuera la original, en cuyo caso se pretende buscar bajo qué condiciones se presenta y en que medida se puede sopesar, ya que hay otras técnicas fuera de este trabajo de investigación para hacer frente a este problema.

1.3. METODOLOGÍAS

Se pretende diseñar un arreglo de dos micrófonos, del cual se debe determinar el tipo de micrófonos y la distancia óptima entre ellos, las señales generadas por estos micrófonos son recibidas por un DSP de la familia C6000.

Actualmente existen varios métodos para efectuar la localización, sin embargo, en esta investigación revisan aquellos métodos que requieren de pocos micrófonos, entre los cuales se pueden mencionar los siguientes:

- **Familia de Métodos Correlación Cruzada Generalizada (GCC).** Estos métodos correlacionan los espectros de las señales de ambos micrófonos, estimando la diferencia de tiempo de arribo y el retraso de tiempo que maximiza la correlación. En la Figura 1.1 se observa la incidencia de un frente de ondas en un arreglo de dos micrófonos y mediante la correlación cruzada se estima el retraso τ, que es la diferencia de tiempo en que incide la onda en el micrófono uno y dos.

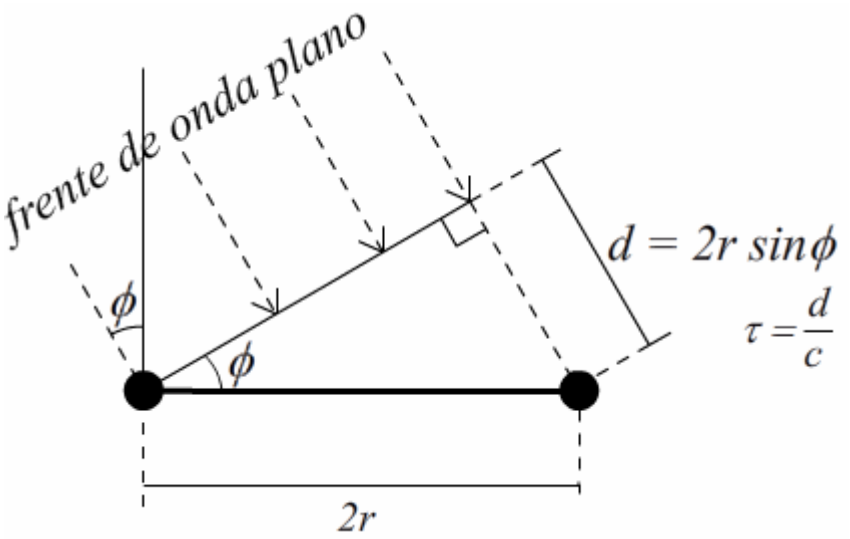


Figura 1.1. Arreglo de dos micrófonos sobre el cual incide un frente de onda plano.

- **Algoritmo Descomposición de Eigenvalores Adaptables.** Se calcula la matriz de covarianza entre las señales de los micrófonos. Se obtiene el eigenvector normalizado correspondiente al eigenvalor más pequeño diferente de cero. Lo anterior mediante algoritmo de la Media de los Mínimos Cuadrados LMS [8, 17].

- **Familia de Métodos por Head Related Transfer Function (HRTF).** Busca aproximar la forma en que el humano efectúa la localización de sonidos, aprovechando su tolerancia al ruido y la habilidad de localizar sonidos en espacios 3D. La aproximación la hace mediante filtros que dependen de la dirección de la fuente de sonido así como efectos de difracción y refracción del cuerpo humano [3]. En la Figura 1.2.a se aprecia un bosquejo de un maniquí con el que comúnmente se obtienen las señales para obtener las funciones de transferencia; en 1.2.b se observa que las señales obtenidas en los micrófonos internos del maniquí, se consideran señales filtradas por el medio de propagación y cuyas funciones de transferencia H_R y H_L pueden ser estimadas con el conocimiento de la señal de la fuente, S.

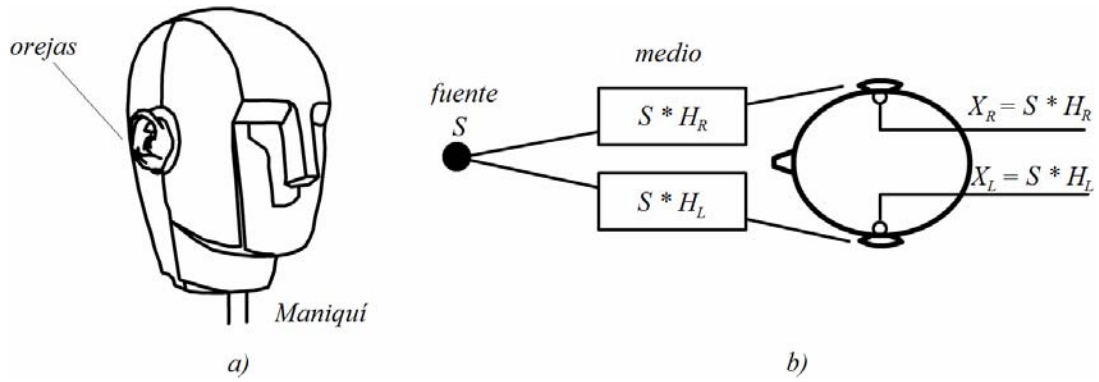


Figura 1.2. a) Maniquí de pruebas y b) Modelo de señales

- **Selección de la Región para Localización de Sonido.** Este es un sistema en el cual se ocupa un arreglo de 3 micrófonos distribuidos en los vértices de un triángulo equilátero. Con los tres micrófonos (*L* left, *R* right y *B* back) se forman tres arreglos. Se calcula la correlación cruzada de las señales de los tres arreglos con lo que se obtienen tres señales las correlaciones cruzadas correspondientes a cada arreglo. Cada arreglo puede hacer la localización en distintas regiones *L-R* entre 60° a 120° y 240° a 300°, *R-B* entre 120° a 180° y 300º a 360°, y por último *L-B* entre 180° a 240° y 0° a 60°, tal como se aprecia en la Figura 1.3. Se comparan las tres señales correlaciones cruzadas y aquella donde se presente el valor máximo de las tres se obtiene el retraso donde se maximiza y se obtiene el Tiempo Estimado de Retraso TDE con lo que se puede calcular la dirección de la fuente de sonido [16,18].

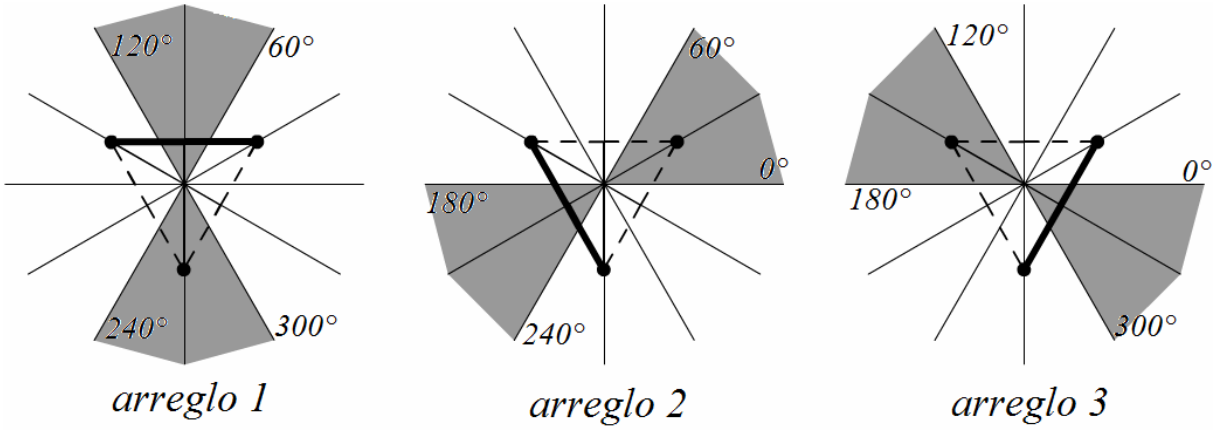


Figura 1.3. Arreglo de tres micrófonos y regiones correspondientes a cada sub arreglo.

- **Método GCC Sumado.** Contiene las mismas características físicas del método anterior. Se realiza la correlación cruzada de las señales de los tres arreglos de acuerdo al método GCC. Una vez obtenidas las correlaciones, estas corresponden al dominio del tiempo, así que se mapean para que se encuentren en el dominio del espacio (referidos a un ángulo de arribo). Una vez mapeadas, las señales se suman. Se obtiene la posición angular de la fuente donde se maximiza la suma de las correlaciones cruzadas. Es posible realizar la localización en un espacio 3D para ello se deben hacer mapeos de las funciones

correlación cruzada tanto del ángulo azimuth ϕ como elevación θ, considerando un sistema de coordenadas esféricas, como se muestra en la Figura 1.4, además de que el arreglo debe presentar una inclinación conocida [16].

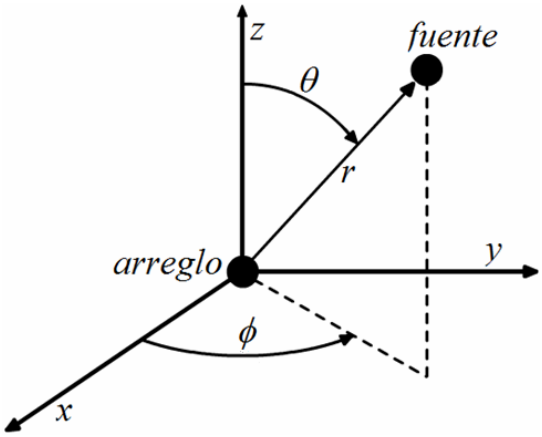


Figura 1.4. Sistema de coordenadas esféricas.

De estos métodos se debe elegir el óptimo en el sentido de que requiera de pocos micrófonos el arreglo, así como el proceso de las señales no sobrepase las capacidades del dispositivo a utilizar, en este caso una arquitectura DSP, es por ello que los más viables, son los de selección de región para localización de sonido y método GCC Sumado.

Actualmente existen varios métodos y algoritmos para realizar la localización de fuentes de sonido mediante micrófonos.

Para ello se debe construir un arreglo de tres micrófonos ubicados sobre un triangulo equilátero en primera instancia con micrófonos electret em-926 ya que son de bajo costo, de dimensiones adecuadas para el proyecto, con un diámetro de nueve milímetros del cual se obtendrán las señales, que serán procesadas por una arquitectura DSP de Texas Instruments de la familia C6000.

Las señales se correlacionan entre sí y de los tres sub arreglos que se forman se estimara el tiempo de retraso entre las señales a partir del cual se obtiene un estimado del ángulo al cual arriba la onda proveniente de la fuente de sonido, con lo cual se haría la localización de la fuente de sonido, lo que consistiría en hacer un estimado del ángulo de azimuth y elevación.

1.4. DESCRIPCION DE CAPÍTULOS

2. Análisis de señales y percepción acústica. Se da un breve repaso de los conceptos básicos del análisis de señales, que son necesarios para entender el algoritmo elegido; así como los fenómenos físicos que se presentan en el proceso de detección, las cuales a su vez, pueden representar dificultades y restricciones en dicho proceso.

3. Algoritmos y métodos para la localización de fuentes de sonido. Se presentan algunos métodos recientes, que se han desarrollado para la detección de fuentes de sonido,

enfocándose en aquellos que requieren un número mínimo de micrófonos, dando un mayor énfasis aquellos basados en Correlación Cruzada Generalizada GCC. Con el objetivo de dar un panorama general del estado actual del tema. También Se hace una breve exposición del porque se eligieron estos dos métodos como objeto principal de investigación, así como los fundamentos teóricos en los que se basan y las ventajas y restricciones que presentan.

4. Diseño e implementación del sistema. Se presentan la arquitectura de procesamiento elegida para realizar la detección, es decir los DSPs de Texas Instruments TMS320F2812 y el TMS320C6416, la construcción y diseño del arreglo de micrófonos empleado, y el diseño del programa de procesamiento de las señales, en el cual se encuentra el algoritmo.

5. Pruebas y resultados. Descripción de los experimentos y las condiciones en las cuales se desarrollaron los experimentos.

6. Conclusiones sobre los resultados finales obtenidos.

- Página en blanco intencionalmente -

2

ANÁLISIS DE SEÑALES Y PERCEPCIÓN ACÚSTICA

En este capítulo se presentan los conceptos básicos del análisis de señales, tales como definición de señal, obtención del espectro de una señal, cálculo de la correlación de señales, eigenvalores y eigenvectores, para entender los métodos expuestos en el capitulo 3; así como conceptos de acústica, la forma en que se propagan las ondas sonoras, los fenómenos que se presentan al propagarse en un medio, tales como reverberación, difracción, reflexión y refracción. Para dar a conocer los conceptos necesarios que se utilizan en los algoritmos, además de las restricciones físicas que se tienen en el experimento.

Ya que la correlación es la base del algoritmo que se utiliza y se revisa en los capítulos 3 y 4 es el primer concepto que se expone.

2.1. SEÑALES DE TIEMPO CONTINUO Y TIEMPO DISCRETO

Las señales son representaciones de fenómenos tales como el cambio de presión atmosférica, el desplazamiento de un objeto, los cambios de presión en el aire debido a un objeto que esta vibrando. Estas representaciones matemáticas son funciones de una o más variables independientes [3].

En general, considerando la naturaleza de las variables independientes, en especial el tiempo, hay una forma de clasificar a las señales: las señales tiempo continuo y las señales tiempo discreto. Una señal de voz es una señal de tiempo continuo, es decir, la variable independiente que se utiliza para su representación es continua. Por otro lado están las señales tiempo discreta, las cuales solo están definidas en tiempos discretos, por lo que la variable independiente toma solo una serie de valores. Para distinguir entre señales tiempo continuo y tiempo discreto se utiliza el símbolo *t* para la variable tiempo continuo y *n* para la variable tiempo discreto

2.1.1 Adquisición de una señal tiempo discreto a partir de una señal tiempo continuo

Una señal discreta se puede adquirir mediante la toma de muestras de la señal continua, lo cual se hace mediante convertidores analógico digitales, pero considerando el teorema del muestreo, para que la señal obtenida represente adecuadamente a la señal continua [11].

2.2. AUTOCORRELACIÓN Y CORRELACIÓN CRUZADA

La función autocorrelación de un proceso aleatorio *X(t)* es la correlación $E[X_1X_2]$ de dos variables aleatorias, $X_1 = x(t_1)$ y $X_2 = x(t_2)$, definidas por el proceso *X* en los tiempos t_1 y t_2. La ecuación (2.1) define la autocorrelación, donde *E* significa esperanza matemática.

$$R_{XX}(t_1, t_2) = E[X(t_1)X(t_2)] \tag{2.1}$$

Lo cual, si se realiza la asignación $t_1 = t$ y $t_2 = t_1 + \tau$, con el desplazamiento real en el tiempo τ, se tiene (2.2)

$$R_{XX}(t, t+\tau) = E[X(t)X(t+\tau)] \tag{2.2}$$

La función de correlación cruzada de dos procesos aleatorios *X(t)* y *Y(t)* está definida por la ecuación (2.3)

$$R_{XY}(t, t+\tau) = E[X(t)Y(t+\tau)] \tag{2.3}$$

Como se puede observar de las expresiones (2.1) y (2.3), sólo se pueden calcular si se conocen las funciones densidad de probabilidad de los procesos aleatorios, lo cual en las funciones densidad de probabilidad no se conocen [17].

2.2.1. Estimación de la función correlación cruzada

Si se considera el producto de *f(t)* y *g(t)*, desplazando en el tiempo τ una de las funciones por ejemplo *f(t)*, la expresión formada es una función del desplazamiento del tiempo. Esto lleva a la definición de la función correlación cruzada que está dada por la ecuación (2.4)

$$\varphi_{fg}(\tau) = \int_{-\infty}^{\infty} f(t+\tau)g^*(t)dt \tag{2.4}$$

De la cual se desprende la función auto correlación, haciendo en la ecuación (2.5) *f(t) = g(t)*

$$\varphi_{ff}(\tau) = \int_{-\infty}^{\infty} f(t+\tau)f^*(t)dt \tag{2.5}$$

La función correlación cruzada describe cuan relacionadas están las dos señales. La función auto correlación muestra cuan similares son las componentes de una señal de tiempo que

ocurren en diferentes puntos del tiempo. Esto es de utilidad, ya que es la base en el método GCC y GCC sumado que se explica en capitulo 3.

La función correlación cruzada para señales determinísticas puede ser definida por una convolución, ecuación (2.6).

$$\varphi_{fg}(\tau) = \int_{-\infty}^{\infty} f(t+\tau) g^{*}(t) dt = f(\tau) * g^{*}(-\tau) \tag{2.6}$$

Por lo que la autocorrelación está dada por la ecuación (2.7)

$$\varphi_{ff}(\tau) = \int_{-\infty}^{\infty} f(t+\tau) f^{*}(t) dt = f(\tau) * f^{*}(-\tau) \tag{2.7}$$

2.3. TRANSFORMADA DISCRETA DE FOURIER Y TRANSFORMADA RÁPIDA DE FOURIER FFT

El método de transformada de fase PHAT hace uso de la transformada de Fourier como se explica en el capítulo 3. La obtención por radix 2 es la forma elegida para calcular la FFT, como se explica en el capítulo 4, por ello es necesario hacer una revisión de estos conceptos. La transformada discreta de Fourier de una señal *x[n]*, de N puntos [11], denotada como *X(k)* se define como

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-jk(2\pi/N)n} \quad k = 0, 1, \ldots, N - 1 \tag{2.8}$$

A su vez, a partir de la DFT *X(k)* se puede volver a obtener x[n] con la ecuación (2.9)

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X(k) e^{jk(2\pi/N)n} \quad n = 0, 1, \ldots, N - 1 \tag{2.9}$$

Para el cálculo de la DFT, considerando que *x[n]* es compleja, se requiere de un total de N^2 multiplicaciones complejas y *N(N - 1)* adiciones complejas. Se puede inferir que el cálculo de la DFT de una señal puede ser un proceso pesado y llevar mucho tiempo de procesamiento, por el número de operaciones a efectuar. Por lo que son necesarios procedimientos para reducir el número de multiplicaciones y adiciones. Los algoritmos que buscan calcular la DFT de forma más eficiente reciben el nombre de algoritmos Transformada Rápida de Fourier FFT [11].

En la ecuación (2.10) se tiene el exponencial

$$W_N = e^{-j\frac{2\pi}{N}} \tag{2.10}$$

Los cuales explotan la simetría y periodicidad del término exponencial

Al elevar (2.10) a la potencia *nk*, entonces se tiene (2.11)

$$W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \tag{2.11}$$

Entonces

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad k = 0, 1, \ldots, N - 1 \tag{2.12}$$

Y su inversa

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) W_N^{-kn} \quad n = 0, 1, \ldots, N - 1 \tag{2.13}$$

$$W_N^{k[N-n]} = W_N^{-kn} = (W_N^{kn})^* \text{ simetría conjugada compleja} \tag{2.14}$$

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n} \text{ periodicidad en } n \text{ y } k \tag{2.15}$$

Uno de estos procesos explota la simetría y periodicidad de la exponencial compleja, y la secuencia x[n] se descompone en secuencias más pequeñas, son llamados algoritmos decimación en el tiempo.

El caso más sencillo es el caso de *N* igual a un entero potencia de 2, $N = 2^v$.

Dadas la ecuaciones (2.11) y (2.12), se separa *x[n]* en puntos pares e impares y se obtiene (2.16)

$$X(k) = \sum_{npares} x[n] W_N^{nk} + \sum_{nimpares} x[n] W_N^{nk} \tag{2.16}$$

Con la sustitución de variables $n = 2v$ para *n* par y $n = 2r + 1$ para *n* impar

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_N^{2rk} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_N^{(2r+1)k} \tag{2.17}$$

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x[2r] (W_N^2)^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] (W_N^2)^{rk} \tag{2.18}$$

Dado

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi/(N/2)} = W_{N/2} \tag{2.19}$$

Se reescribe

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_{N/2}^{rk} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_N^{rk} \tag{2.20}$$

$$X(k) = G(k) + W_N^k H(k) \quad \text{k = 0, 1, …, N – 1} \tag{2.21}$$

Esta descomposición entre puntos pares e impares se hace de forma iterativa hasta obtener el cálculo de términos de dos muestras. De esta forma el número de multiplicaciones complejas y sumas es igual a $Nv = Nlog_2N$

Por ejemplo, para el caso de *x[n]* de 8 muestras, se obtiene la siguiente estructura que representa el cálculo de la FFT mostrada en la figura 2.1.

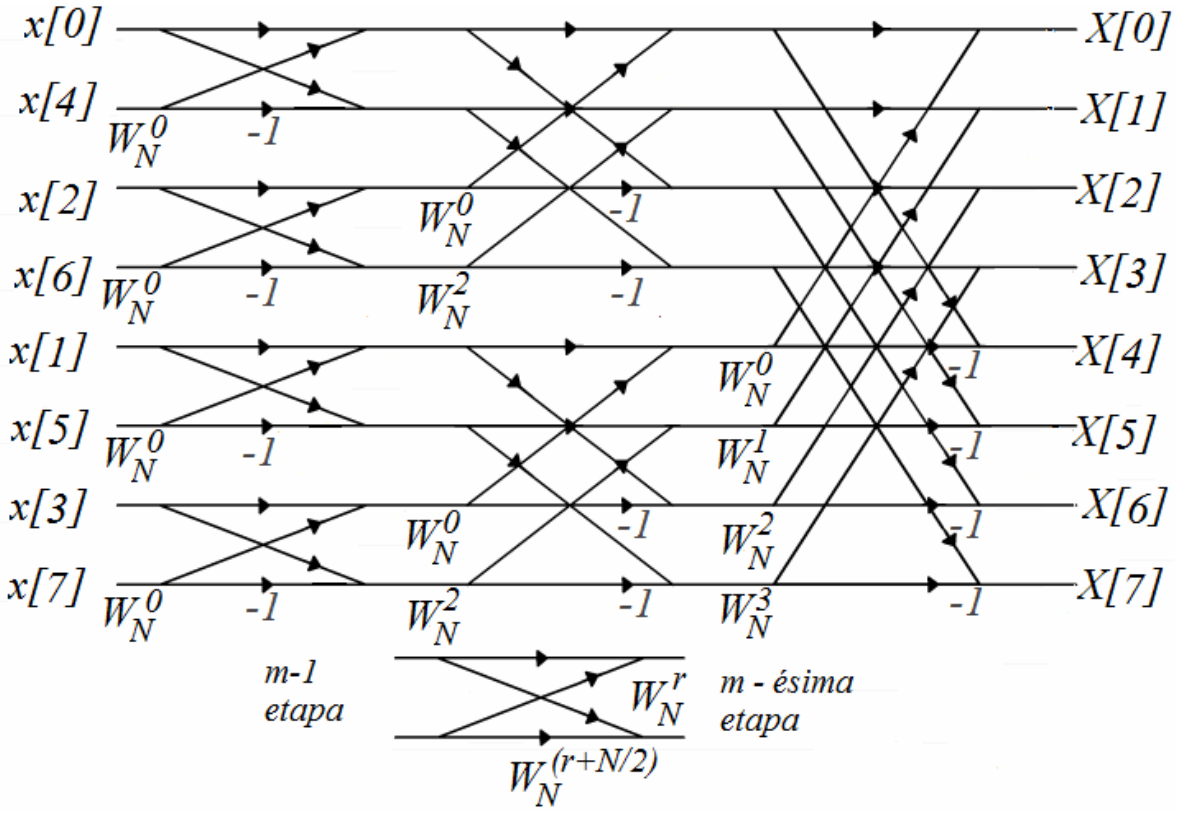


Figura 2.1. Transformada Rápida de Fourier radix 2 de una secuencia de 8 muestras.

Al fondo de la Figura 2.1 se observa la estructura básica llamada mariposa.

2.4. FFT INVERSA

Utilizando la misma red de FFT radix 2 decimada en tiempo se puede calcular la FFT Inversa siguiendo el procedimiento

1. Introducir *X(k)* decimada.

2. En la red, cambiar los coeficientes W^n en el orden : $W_N^0 = 1$ Y el resto de coeficientes W cambiar el signo $-W_N^n$

3. Dividir la secuencia de salida entre N.

2.5. FUNCIÓN CORRELACIÓN CRUZADA Y ESPECTRO POTENCIA CRUZADA

Considerando las transformadas de las señales definidas de $-T$ a T mostradas en las ecuaciones (2.22) y (2.23)

$$X_T(\omega) = \int_{-T}^{T} x(t) e^{-j\omega t} dt \tag{2.22}$$

$$Y_T(\omega) = \int_{-T}^{T} y(t) e^{-j\omega t} dt \tag{2.23}$$

En (2.24) se expresa $S_{xy}(\omega)$

$$S_{xy}(\omega) = \lim_{T \to \infty} \frac{E[X_T^*(\omega) Y_T(\omega)]}{2T} \tag{2.24}$$

$$P_{xy} = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_{xy}(\omega) d\omega \tag{2.25}$$

Se toma el valor esperado y el límite como en (2.24) y (2.25) en (2.22) y (2.23) para obtener $S_{xy}(\omega)$

$$X_T^*(\omega) Y_T(\omega) = \int_{-T}^{T} x(t_1) e^{j\omega t_1} dt_1 \int_{-T}^{T} y(t_2) e^{j\omega t_2} dt_2 \tag{2.26}$$

$$X_T^*(\omega)Y_T(\omega) = \int_{-T}^{T}\int_{-T}^{T} x(t_1)y(t_2)e^{-j\omega(t_2-t_1)}dt_1dt_2 \tag{2.27}$$

Cambiando variables acorde a (2.28) (2.29)

$$t = t_1 \quad dt = dt_1 \tag{2.28}$$

$$\tau = t_2 - t_1 = t_2 - t \quad d\tau = dt_2 \tag{2.29}$$

Mediante manipulación algebraica se tiene

$$\frac{E[X_T^*(\omega)Y_T(\omega)]}{2T} = E\left[\int_{-T-t}^{T-t}\left\{\frac{1}{2T}\int_{-T}^{T} x(t)y(t+\tau)dt\right\}e^{-j\omega\tau}d\tau\right] \tag{2.31}$$

$$\frac{E[X_T^*(\omega)Y_T(\omega)]}{2T} = \int_{-T-t}^{T-t}\left\{\frac{1}{2T}\int_{-T}^{T} R_{xy}(t,t+\tau)dt\right\}e^{-j\omega\tau}d\tau \tag{2.32}$$

Ahora se toma el límite como en (2.24)

$$S_{xy}(\omega) = \lim_{T\to\infty}\int_{-T-t}^{T-t}\left\{\frac{1}{2T}\int_{-T}^{T} R_{xy}(t,t+\tau)dt\right\}e^{-j\omega\tau}d\tau \tag{2.33}$$

$$S_{xy}(\omega) = \int_{-T-t}^{T-t}\left\{\lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} R_{xy}(t,t+\tau)dt\right\}e^{-j\omega\tau}d\tau \tag{2.34}$$

Dado (2.34) que es una transformada de Fourier y tal transformación es única, se aplica la transformada inversa

$$\lim_{T\to\infty}\frac{1}{2T}\int_{-T}^{T} R_{xy}(t,t+\tau)dt = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{xy}(\omega)e^{-j\omega\tau}d\omega \tag{2.35}$$

Dado el espectro potencia cruzada, la función correlación cruzada no puede en general ser recuperada, solo su tiempo promedio. Para procesos conjuntamente estacionarios en sentido amplio, la función correlación cruzada $R_{xy}(\tau)$ puede ser encontrada a partir de $S_{xy}(\omega)$ desde su tiempo promedio es solo $R_{xy}(\tau)$.

2.6. EIGENVALORES Y EIGENVECTORES

El algoritmo Descomposición de Eigenvalores Adaptable es un método de efectuar la localización de una fuente de voz el cual se detalla en el capítulo 3, pero antes es necesario tener

los conceptos de eigenvalores y eigenvectores que son la base del algoritmo. Sea A una matriz *nxn* y considere las siguientes ecuaciones

$$Av = \lambda v \tag{2.36}$$

Donde λ es constante. La ecuación anterior puede expresarse de la forma

$$(A - \lambda I)v = 0 \tag{2.37}$$

Para que un vector diferente al vector nulo sea una solución a esta ecuación es necesario que $A - \lambda I$ sea singular. Por lo tanto el determinante de $A - \lambda I$ debe ser *0*

$$p(\lambda) = \det(A - \lambda I) = 0 \tag{2.38}$$

$p(\lambda)$ es el polinomio característico de la matriz *A* y las *n* raíces λ_i, para *i = 1, 2, 3, …n*, son los eigenvalores de *A*.

Dada la ecuación (2.39)

$$Av_i = \lambda_i v_i \tag{2.39}$$

Los vectores v_i son los eigenvectores de *A*, λ_i los eigenvalores por vector v_i diferente del vector nulo [7].

Propiedades [7]

Propiedad 1. Los vectores diferentes de cero v_1, v_2, …, v_n correspondientes a distintos eigenvalores λ_1, λ_2, …, λ_n, son linealmente independientes

Propiedad 2. Los eigenvalores de una matriz hermitiana son reales.

Propiedad 3. Una matriz hermitiana es definida positiva $A > 0$, si y solo si los eigenvalores de *A* son positivos, $\lambda_k > 0$.

Propiedad 4. Los eigenvectores de una matriz hermitiana correspondientes a distintos valores son ortogonales, es decir, si $\lambda_i \neq \lambda_j$ entonces $< v_i, v_j >= 0$

Teorema espectral. Cualquier matriz hermitiana puede descomponerse como

$$A = V\Lambda V^H = \lambda_1 v_1 {v_1}^H + \lambda_2 v_2 {v_2}^H + ... + \lambda_n v_n {v_n}^H \tag{2.40}$$

Donde λ_i y vi son los eigenvalores y eigenvectores de *A*.

Propiedad 5. Sea *B* una matriz de *nxn* con eigenvalores λ_i y sea *A* una matriz que esté relacionada con *B* de acuerdo a

$$A = B + \alpha I \tag{2.41}$$

Entonces *A* y *B* tienen los mismos eigenvectores y los eigenvalores de *A* son $\lambda_i + \alpha$

Propiedad 6. Para una matriz *A* simétrica positiva la ecuación

$$x^T Ax = 1 \tag{2.42}$$

Define una elipse en n dimensiones cuyos ejes están en la dirección de los eigenvectores *vj* de *A* con la mitad de longitud de estos ejes igual a $\frac{1}{\sqrt{\lambda_j}}$

Propiedad 7. El eigenvalor más grande de una matriz *nxn* $A = \{a_{ij}\}$ está limitada

$$\lambda_{\max} \leq \max_i \sum_{j=1}^{n} a_{ij} \tag{2.43}$$

2.7. FUNDAMENTOS DE ACÚSTICA

Para la transmisión de sonido se requieren tres elementos: una o varias fuentes, un receptor, y el medio de propagación. La transmisión parte de la existencia de partículas cuya posición puede ser modificada. Dependiendo del movimiento de las partículas y la dirección de la onda se hace una clasificación que a continuación se explica.

2.7.1. Mecanismos de propagación de ondas

Las ondas pueden ser de dos tipos:

- Ondas transversales: el desplazamiento de las partículas del medio es perpendicular a la dirección de propagación de la onda, figura 2.2.a.

- Ondas longitudinales: el desplazamiento de las partículas es paralelo a la dirección de propagación de la onda, figura 2.2.b.

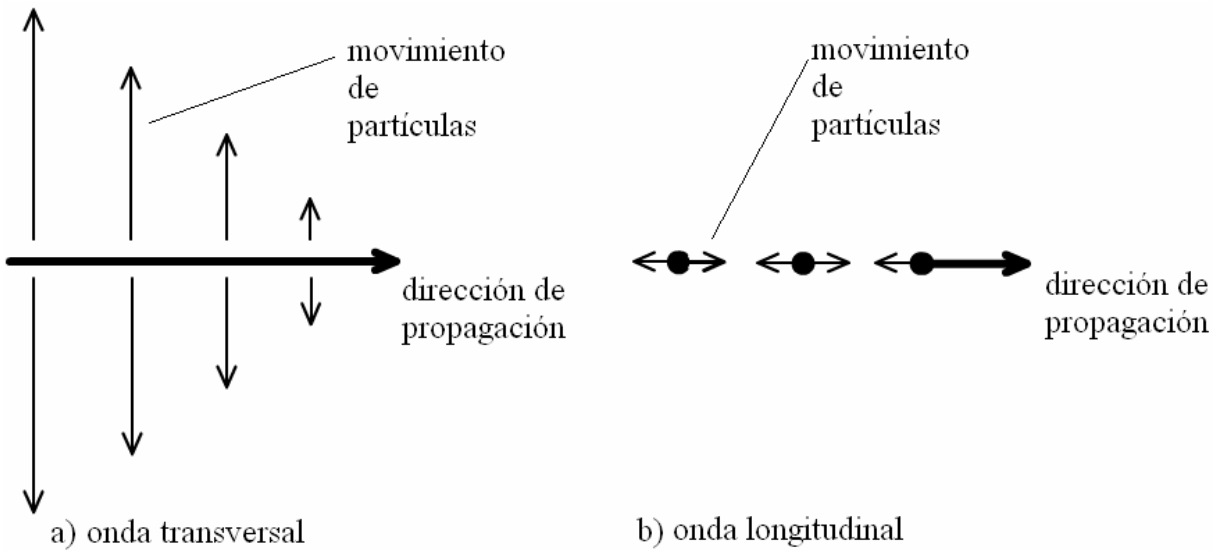


Figura 2.2. Tipos de onda.

En los sólidos, las ondas sonoras son ondas compuestas, con una componente transversal y una componente longitudinal. En los líquidos al igual que en los sólidos la onda es compuesta aunque la componente longitudinal es predominante. Y en los gases las vibraciones transversales son casi despreciables predominando la componente longitudinal [2].

2.7.2. Reflexión, refracción y difracción del sonido

Si la longitud de onda del sonido es pequeña en comparación a las dimensiones de una superficie lisa y plana, el sonido es reflejado al igual que la luz. Considerando la similitud con la luz, existe un ángulo de incidencia del sonido sobre la superficie plana y el cual es igual al ángulo de reflexión.

La dirección de la onda de sonido al igual que la luz cambia su dirección de propagación en lineal recta si el medio es no homogéneo. La refracción y difracción son nombres dados a dos fenómenos que provocan que cambie la dirección de la onda. La refracción es debida a cambios espaciales en la velocidad de propagación en el medio. Para el sonido estas variaciones espaciales son debidas a cambios de presión ambiental, cambios de temperatura, y cambios de densidad.

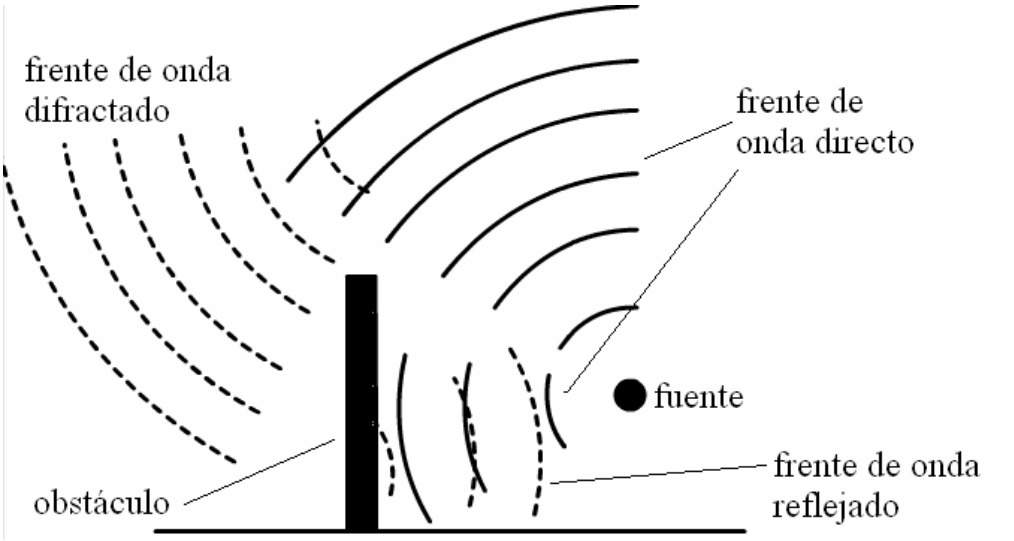


Figura 2.3. Frente de onda directo, reflejado y difractado.

Cuando el sonido incide sobre un objeto el cual es más largo en comparación a la longitud de onda del sonido, se forma un efecto sombra, pero parte del sonido es dirigido dentro del área de sombra por lo que se llama difracción, como se observa en la Figura 2.4.

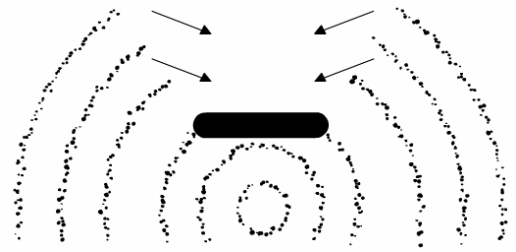


Figura 2.4. Efecto sombra, obstáculo mayor a la longitud de onda.

La difracción describe al fenómeno de las ondas que se doblan alrededor de los objetos y representa un desvío del modelo geométrico de la propagación de la onda. La difracción explica porque los bordes de las sombras no están bien definidos y porque se pueden escuchar alrededor de las esquinas [2].

Si una barrera tiene un agujero pequeño en comparación a la longitud de onda del sonido, el agujero esencialmente se convierte en una fuente puntual, radiando sonido con frentes de onda esféricos dentro de la región tras la barrera. Como se observa en la Figura 2.5.

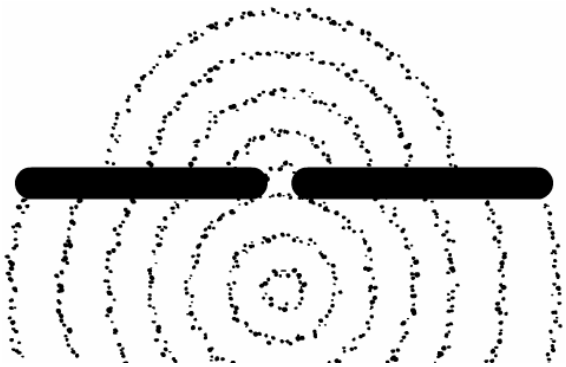


Figura 2.5. Difracción abertura menor a la longitud de onda.

2.7.3. Reverberación

Cuando la energía del sonido es introducido en un recinto, el nivel de sonido aumenta hasta un nivel estable sobre un periodo de tiempo usualmente entre 0.25 *s* a 15 *s*. Cuando las fuentes de sonido cesan, entonces el sonido gradualmente decae alrededor de un tiempo similar. A este tiempo se le llama tiempo de reverberación y se define como el tiempo para que el sonido decaiga por 60 *dB*. Esta caída de 60 *dB* es casi igual al tiempo tomado para un nivel de voz alta (cerca de 80 *dB*) para decaer hasta perderse en el fondo de un salón silencioso (cerca de 20 *dB*). El tiempo de reverberación depende del tamaño del salón, y la extensión en la cual el sonido es absorbido por los muros, muebles, etc. [2]. El cálculo del tiempo de reverberación puede hacerse por la formula Sabine (2.44) [2].

$$RT = \frac{0.16V}{A} \tag{2.44}$$

Donde *RT* es el tiempo de reverberación en segundos, *V* es el volumen del salón en metros cúbicos y *A* la absorción del salón total en *sabins* (m^2).

La absorción total se calcula por la suma de contribuciones de todas las superficies absorbentes.

$$A = S_1\alpha_1 + S_2\alpha_2 + S_3\alpha_3 + ... + S_n\alpha_n \tag{2.45}$$

S_n es el área de la superficie, a_n el coeficiente de absorción de la superficie *n*.

El tiempo de reverberación tiene efectos importantes en la inteligibilidad de la voz, y en la calidad de sonido de la música. En el caso de la voz, un tiempo corto de reverberación implica alta absorción, lo cual se refleja en una dificultad para un orador proyectar su voz en un nivel suficiente para alcanzar los asientos que están mas al fondo del salón.

Por otro lado, un tiempo largo de reverberación produciría que cada silaba pronunciada sería escuchada pero se sobrepondrían por las sílabas anteriores por lo que se perdería la inteligibilidad. Para una inteligibilidad máxima se requiere un tiempo de reverberación de no mas

de un segundo y tiempos superiores de dos segundos crean confusión en el auditorio ya que no se podría percibir con precisión cada sílaba.

2.7.4. Superposición del Sonido

En un medio como el aire, cuando viajan más de un frente de onda de sonido al mismo tiempo, las partículas de aire responderán a la suma vectorial de los desplazamientos correspondientes a cada frente de onda. Esto corresponde al principio de superposición

2.7.5. Ley cuadrado inverso

El sonido conforme se aleja de la fuente de emisión disminuye su intensidad, para saber como es el cambio en intensidad primero se considera que la intensidad de sonido está definida como la potencia por centímetro cuadrado *W/cm*^{2}. Es decir el número de *watts* dividido por el área de la superficie, es decir el área de la esfera (2.46).

$$I = \frac{W}{4\pi r^2} \tag{2.46}$$

Donde *I* es intensidad, *W* es potencia en *watts* y *r* la distancia desde la fuente al frente de onda en *m*^{2}. Ahora bien, considerando una distancia r_1 y una distancia r_2, como se muestra en la Figura 2.6 se tiene

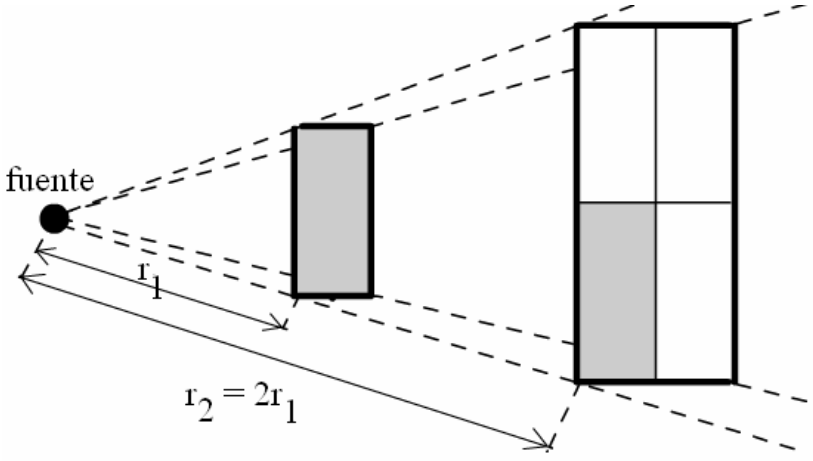


Figura 2.6. Frente de onda plano que se desplaza el doble de la distancia con lo que la potencia se divide en cuatro.

Es la misma potencia del sonido en ambos segmentos esféricos en r_1 y r_2, para obtener la proporción de I_1 e I_2 se obtiene la ecuación

$$\frac{I_1}{I_2} = \frac{4\pi r_2^{\,2}}{4\pi r_1^{\,2}} = \frac{r_2^{\,2}}{r_1^{\,2}} \tag{2.47}$$

A esta ecuación se le llama ley de cuadrados inversos la cual solo aplica en condiciones de campo libre. Dado que la presión de sonido es proporcional a la raíz cuadrada de la intensidad, la presión del sonido en r_2 es la mitad en r_1 o 6 *dB* por debajo lo cual lleva a la declaración que dice que la presión del sonido en un campo libre cae 6 *dB* por el doble de la distancia [15].

2.7.6. Área de Audición Humana

La Figura 2.7 muestra en forma técnica los límites de la percepción aural humana. El área está limitada en niveles de sonido bajo por el umbral de audición. Los sonidos más suaves que pueden ser escuchados caen sobre el umbral de curva auditiva. Sobre esta línea el movimiento de las moléculas del aire es suficiente para obtener una respuesta. Si, en alguna frecuencia dada, el nivel de presión del sonido es incrementado lo suficiente, se alcanza un punto en el cual un cosquilleo es percibido en los oídos. Si el nivel es incrementado considerablemente sobre el umbral de sensación, se vuelve doloroso. Tanto para el umbral bajo y alto del área de audición. Existen límites de frecuencia por debajo de los 20 *Hz* y sobre los 16 *kHz*, limites que varían de individuo a individuo. En el área de audición de la Figura 2.7. Todos los sonidos se encuentran en el área sombreada, bajas o altas frecuencias, muy suaves o muy intensas.

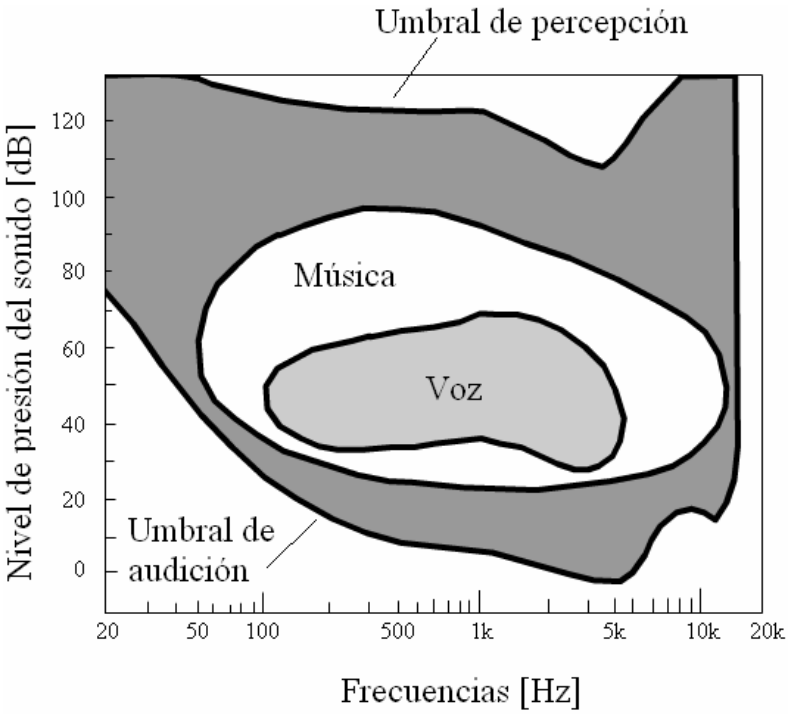


Figura 2.7. Percepción aural humana.

La voz no ocupa el total del área de audición, solo una parte como se muestra en la Figura, su rango dinámico y rango de frecuencia son bastante limitados. La música tiene tanto un gran rango dinámico en comparación de la voz y mayor rango de frecuencias. Pero aún la música no utiliza el área de audición total [3].

2.8. SÍNTESIS

Este capítulo sirve para conocer los conceptos básicos en los cuales se basan los métodos de localización que se mencionan en el capítulo 3 y como se utilizan n el capítulo 4. Se describe la autocorrelación y correlación cruzada que son necesarios para entender el método de localización por el método GCC, y que además son utilizados por las HTRF. La obtención del espectro de una señal con la transformada rápida de Fourier radix 2 es necesario para efectuar la localización por transformada PHAT. También se mencionan lo que son los eigenvalores y eigenvectores que son utilizados por el algoritmo Descomposición de Eigenvalores Adaptable. Todos estos métodos son descritos en el siguiente capítulo.

Dado que las señales se obtienen ondas sonoras es necesario conocer como se propaga el sonido en el espacio y saber que fenómenos se relacionan con el mismo, tales como la reflexión refracción, difracción superposición del sonido, el cambio de intensidad de sonido conforme se aleja de la fuente. Así como el sonido es percibido por el ser humano.

- Página en blanco intencionalmente -

3

ALGORITMOS Y MÉTODOS PARA LA LOCALIZACIÓN DE FUENTES DE SONIDO

En este capítulo se expone la localización de fuente de sonido, los métodos y algoritmos que existen para realizar la localización enfocándose solamente en arreglos de dos o tres micrófonos ya que hay otros métodos que implican el uso de más micrófonos, con los inconvenientes de requerir procesar un número mayor de señales, que en ciertas aplicaciones pueden ser llegar a ser limitantes.

Se define los modelos ideal de campo libre y real reverberante que son parte del estado del arte y dependiendo del modelo se han obtenido métodos que de acuerdo a las condiciones pueden aplicarse. Se presentan las familias de métodos Correlación Cruzada Generalizada GCC, Algoritmo Descomposición de Eigenvalores Adaptable, Head Related Transfer Functions HRTFs y dos métodos híbridos de GCC, División de Región y GCC Sumado, el cual es el método elegido para hacer la localización y se implementa en el capítulo 4.

3.1. LOCALIZACIÓN DE FUENTES DE SONIDO SSL Y MODELOS

La localización de fuente de sonido representa un problema complejo, en el cual hay que considerar la naturaleza del sonido. El sonido al propagarse presenta varios fenómenos tales como refracción, difracción, difusión, reflexión, reverberación e interferencias. El amplio espectro de las frecuencias del sonido también representa dificultades ya que el sonido de banda ancha puede percibirse de distintas formas dependiendo de las características geométricas del ambiente de propagación [16].

La localización de fuentes de sonido, por sus siglas en ingles SSL, es la determinación de la dirección de un emisor, lo cual incluye la distancia hacia este. La dirección puede expresarse por dos ángulos polares azimuth y elevación. La determinación de la distancia de una fuente de sonido puede obtenerse al medirse la intensidad de sonido y/o su espectro; aunque el conocimiento previo es necesario sobre las características de radiación de la fuente.

Para comprender como se percibe el sonido que se propaga por un medio y es captado por un receptor, se han propuesto dos modelos con los cuales se han desarrollado distintas metodologías. Estos modelos consideran una señal que es captada por un arreglo de sensores que en el caso del sonido son micrófonos. Estos modelos son el modelo ideal de campo libre y el modelo real reverberante que a continuación se describen.

3.1.1. Modelo Ideal de Campo Libre

Este modelo considera que en el medio no se presentan objetos que provoquen reflexiones de onda, ni el efecto de reverberación y solo se considera la atenuación de la señal por pérdidas de propagación la cual viene expresada en la ecuación (3.1).

$$x_n(k) = \alpha_n s(k - \tau_n) + b_n(k) \;\; para\ n = 1, 2, 3, \ldots, N \tag{3.1}$$

Donde *s* es la señal de la fuente, α_n es la atenuación por pérdidas de propagación, τ_n tiempo de propagación y $b_n(k)$ es el ruido aditivo. El ruido aditivo en este caso se supone con media cero, ruido blanco, proceso gaussiano y que no está correlacionado con la señal de la fuente.

3.1.2. Modelo Real Reverberante

Este modelo considera a la señal que se propaga en un medio en el cual su espectro es modificado por él, y consiste en una respuesta al impulso acústica, por conveniencia se considera como un filtro FIR, así la señal recibida en el arreglo de micrófonos está dada por la ecuación (3.2).

$$x_n(k) = h_n * s(k) + b_n(k) \;\; para\ n = 1, 2, 3, \ldots, N \tag{3.2}$$

Donde h_n es la respuesta al impulso del n-ésimo canal, como se puede observar se realiza la convolución lineal (*) con la señal de la fuente *s(k)*, se observa que el termino τ_n no está presente en la ecuación (3.2) ya que es un parámetro oculto. Para entender mejor la ecuación (3.2), se describen los vectores y matrices en las ecuaciones (3.3), (3.4), (3.5) y (3.6).

$$x_n(k) = [x_n(k) \quad \ldots \quad x_n(k - L + 1)]^T \tag{3.3}$$

$$s(k) = [s(k) \quad s(k-1) \quad \ldots \quad s(k - L + 1) \quad \ldots \quad s(k - 2L + 2)]^T \tag{3.4}$$

$$b_n(k) = [b_n(k) \quad \ldots \quad b_n(k - L + 1)]^T \tag{3.5}$$

$$h_n(k) = \begin{bmatrix} h_{n,0} & \dots & h_{n,L-1} & \dots & 0 \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & \dots & h_{n,0} & \dots & h_{n,L-1} \end{bmatrix} \quad (3.6)$$

3.1.3. Diferencia de Tiempo de Arribo (TDOA)

Muchos métodos utilizan lo que se llama la Diferencia de Tiempo de Arribo (TDOA) para efectuar la localización de una fuente, la diferencia es el tiempo de arribo que existe entre el i-ésimo y j-ésimo micrófono de un arreglo, la cual se define mediante la ecuación (3.7) y a partir del mismo existe gran variedad de algoritmos y métodos para detectar la dirección de arribo de una fuente de voz.

$$\tau_{ij} = \tau_i - \tau_j \text{ para } i, j = 1, 2, 3, \dots, N \quad (3.7)$$

A continuación se describen los métodos que se han desarrollado para realizar la detección de dirección de arribo de fuentes de voz haciendo especial énfasis en aquellos métodos que consideran el efecto de reverberación presentes en condiciones normales.

3.2. MÉTODOS QUE UTILIZAN FORMADOR DE HAZ BEAMFORMING

Estos métodos realizan la detección mediante un barrido del espacio utilizando un haz virtual formado con las señales de salida de un arreglo de micrófonos, la forma mas sencilla de este es realizar la suma de todas las señales de los micrófonos obteniendo una señal resultado cuya potencia es muy alta en caso de que la fuente de la señal que incide sobre los micrófonos está directamente enfrente del arreglo [1].

Pueden localizar una fuente de voz y extenderse a más de una fuente en caso de que existan más de una. Para poder obtener el haz virtual es necesario que el frente de onda de la señal que incide sobre el arreglo sea lo suficientemente plano, para lo cual la fuente esté lo suficientemente alejado del arreglo, lo cual en muchas ocasiones la fuente de señal no cumple. Ya que es necesario que la fuente de la señal está alejada del arreglo, la potencia de la señal es menor, y entre mayor sea la distancia hay una mayor degradación de esta debido a la ley me los cuadrados.

No se profundiza en estos métodos ya que requieren de más de tres micrófonos lo cual queda fuera del objetivo de la investigación.

3.3. ALGORITMOS DE ESTIMACIÓN DE TIEMPO DE RETRASO (TDE)

Estos métodos involucran dos pasos de procedimiento, primero se calcula una serie de diferencias de tiempo de arribo TDOA relativos a distintos pares de micrófonos, ya con las

TDOA calculados, además con el conocimiento previo sobre la localización de los micrófonos la localización de la fuente acústica es estimada [1].

Las ventajas de estos algoritmos son que se pueden utilizar tanto en señales con banda ancha como para banda angosta, la resolución puede ser flexible realizando el ajuste de la frecuencia de muestreo y el tamaño del arreglo. El efecto de reverberación del medio solo se considera en el primer paso y por último, con algunos algoritmos recientes pueden ser computacionalmente eficientes.

3.3.1. Algoritmo Correlación Cruzada Generalizado (GCC)

Este algoritmo fue propuesto por Knapp y Carter y está lejos de ser de los más utilizados, este algoritmo calcula aproximaciones del Tiempo de Retraso Estimado TDE, para ello emplea el modelo de campo libre ideal y considera sólo dos micrófonos [4]. Tal como se puede observar en la Figura 3.1.

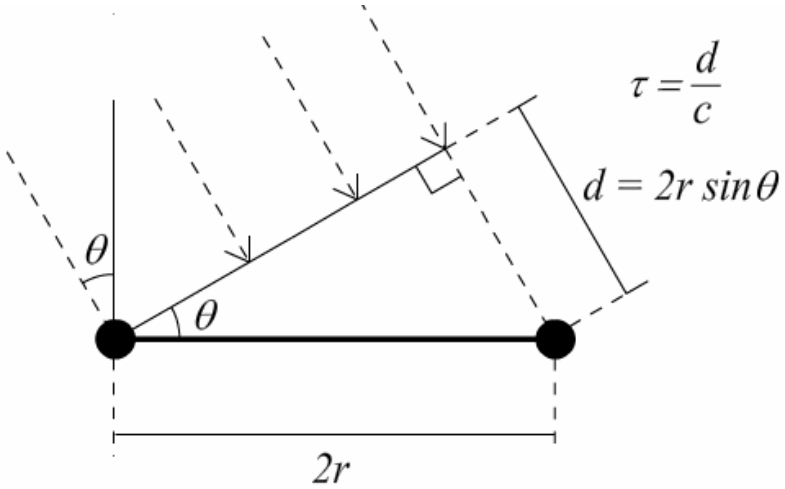


Figura 3.1. Frente de onda plano incidiendo sobre arreglo de dos micrófonos.

Se observa que incide sobre el arreglo de dos micrófonos, separados una distancia *2r*, entre frentes de onda plano y hay un retraso entre señales τ debido al ángulo entre el frente de onda plano y el arreglo dicho retraso está dado por la ecuación (3.8), donde *c* en este caso es la velocidad del sonido

$$\tau = \frac{2r\sin(\theta)}{c} \tag{3.8}$$

Para estimar la TDOA se capturan las señales filtradas obtenidas de los micrófonos, calculando su correlación cruzada, el retraso de tiempo que maximiza la correlación cruzada entre señales es la TDOA estimado. En las ecuaciones (3.10), (3.11) y (3.12) se describe el procedimiento [5, 8,15].

$$\tilde{\tau}_{12} = \arg\max_{\tau} r_{x1x2}{}^{GCC}(\tau) \tag{3.9}$$

Donde

$$r_{x1x2}^{GCC} = F^{-1}\{\psi_{x1x2}(f)\} \tag{3.10}$$

$$r_{x1x2}^{GCC} = \int_{-\infty}^{\infty} \psi_{x1x2}(f) e^{i2\pi f\tau} df \tag{3.11}$$

$$r_{x1x2}^{GCC} = \int_{-\infty}^{\infty} \phi(f) S_{x1x2}(f) e^{i2\pi f\tau} df \tag{3.12}$$

En esto consiste el algoritmo GCC en donde F^{-1} es la transformada inversa discreta de Fourier, $S_{x1x2}(f)$ es la correlación del espectro de las señales x_1 y x_2 como se describe en la ecuación (3.13), el símbolo * significa el valor conjugado.

$$S_{x1x2}(f) = E\{x_1(f) x_2^{*}(f)\} \tag{3.13}$$

El término $\phi(f)$ es una función de pesos en la frecuencia, la cual dependiendo de cómo se calcule da paso a variaciones de este algoritmo; el término dado en (3.11), $\psi_{x1x2}(f)$ se le llama espectro cruzado generalizado. Como se puede observar las correlaciones cruzadas se calculan con los espectros de las señales capturadas por lo que es necesario calcular los espectros.

A continuación se muestran variaciones de este algoritmo para distintos valores de la función de pesos en la frecuencia, lo que da paso a variaciones del método.

3.3.1.a) Método Clásico de Correlación Cruzada (CCC)

Para este algoritmo la función de pesos en al frecuencia se considera igual a 1, es decir $\phi(f) = 1$. Dado el modelo de campo libre ideal (3.14)

$$x_n(f) = \alpha_n e^{-i2\pi f\tau_n} s(f) + B_n(f) \;\; para\ n = 1, 2 \tag{3.14}$$

Se sustituye (3.13) en (3.12) para obtener el espectro cruzado generalizado $\psi_{x1x2}(f)$ tomando en cuenta que $\phi(f) = 1$, con lo que se obtiene las ecuaciones (3.15) y (3.16).

$$\psi_{x1x2}(f) = \phi(f) S_{x1x2}(f) \tag{3.15}$$

$$\psi_{x1x2}(f) = \alpha_1\alpha_2 e^{-i2\pi f\tau_{12}} E\{|S(f)|^2\} \tag{3.16}$$

Ya que $\psi_{x1x2}(f)$ depende de la fuente de señal no es viable su cálculo para obtener el TDE, ya que es necesario que la señal sea estacionaria, dado que la voz es una señal no estacionaria. En al Figura 3.2 se muestra la correlación de dos señales muestreadas a 48 kHz

procedentes de un frente de onda que incide sobre un arreglo de dos micrófonos a un ángulo de 10°.

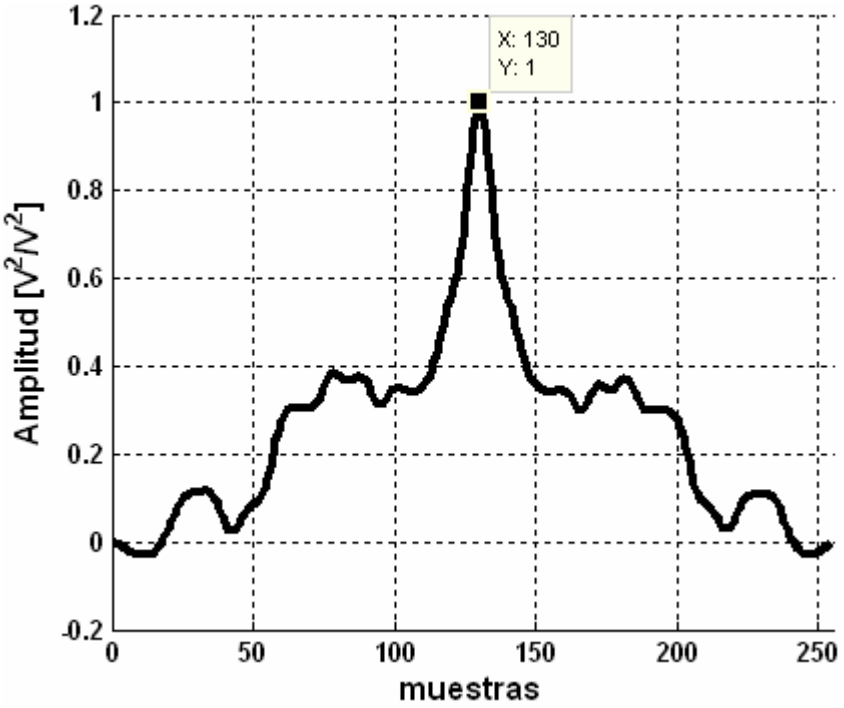


Figura 3.2. Correlación de dos señales.

Al correlacionar dos señales tomando ventanas de 128 muestras se obtiene la señal correlación cruzada que es de 255 muestras, la muestra central es la 128, dado que hay un retraso entre señales el valor máximo se encuentra en la muestra 130, de la señal correlación cruzada, se puede inferir que la muestra central es correspondiente a que no exista retraso entre ambas señales, así que para hacer el mapeo de muestras a grados se toma la ecuación (3.8) y se obtiene (3.17)

$$n = round\left(\frac{2rF_s \sin(\theta)}{c}\right) \tag{3.17}$$

Donde n son las muestras alrededor de la muestra central que ofrecen información sobre el retraso entre señales en términos del ángulo θ, además como se puede observar entre mayor sea la frecuencia de muestreo F_s será mayor la resolución, en este caso la información se encuentra en el intervalo [-24+128, 24+128], para hacer el mapeo de (3.17) se obtiene (3.18).

$$\theta = \arcsin\left(\frac{nc}{2rF_s}\right) \tag{3.18}$$

Que para la muestra n se obtiene el ángulo θ correspondiente así la función mapeada se muestra en la Figura 3.3, como se puede observar en la figura el valor máximo se encuentra a 4.78° que es el valor estimado del ángulo donde se localiza la fuente de sonido.

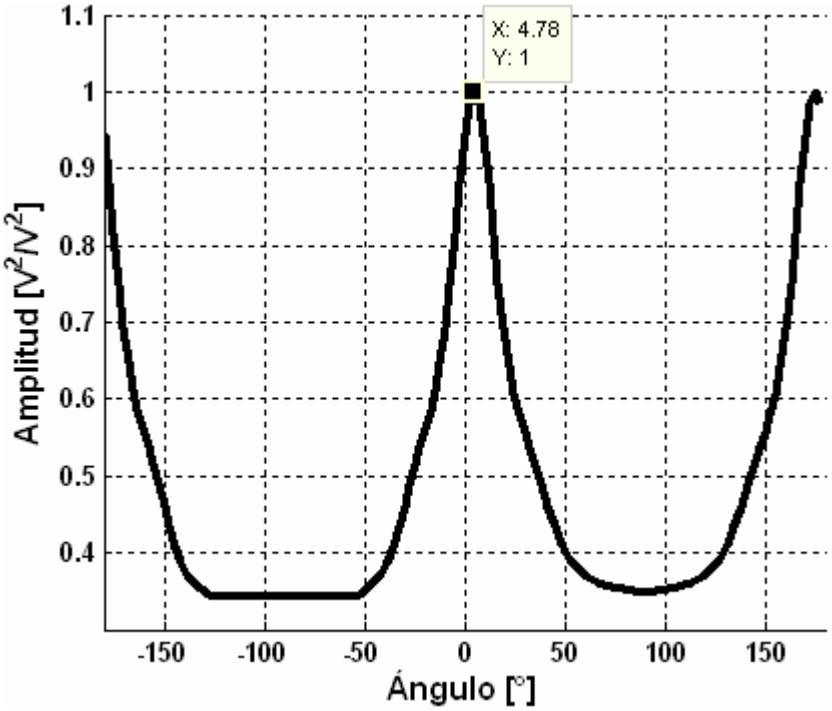


Figura 3.3. Correlación cruzada mapeada en ángulo.

3.3.1.b) Transformada Coherencia Suavizada (SCOT).

Para mitigar la dependencia del espectro cruzado generalizado de la señal de voz para obtener el TDE, una forma de hacerlo es remover autocorrelaciones no deseadas de las señales de los micrófonos antes de calcular el espectro cruzado para esto la función de pesos espectral se hace igual a (3.19), a lo que le llama preblanqueado (prewhitening)[4, 16].

$$\phi(f) = \frac{1}{\sqrt{E\left\{\left|x_1(f)\right|^2\right\}E\left\{\left|x_2(f)\right|^2\right\}}} \tag{3.19}$$

De (3.19) se obtiene el método de Transformada coherente suavizada o SCOT, por lo que el cálculo del espectro cruzado se hace de acuerdo a (3.22), sustituyendo (3.19) en (3.15).

$$\psi_{x1x2}{}^{SCOT}(f) = \frac{\alpha_1\alpha_2 e^{-i2\pi f\tau_{12}} E\left\{\left|S(f)\right|^2\right\}}{\sqrt{E\left\{\left|x_1(f)\right|^2\right\}E\left\{\left|x_2(f)\right|^2\right\}}} \tag{3.20}$$

Desarrollando (3.20) se obtiene:

$$\psi_{x1x2}{}^{SCOT}(f) = \frac{\alpha_1\alpha_2 e^{-i2\pi f\tau_{12}} E\left\{\left|S(f)\right|^2\right\}}{\sqrt{\alpha_1{}^2 E\left\{\left|S(f)\right|^2\right\}+\sigma_{b1}{}^2(f)}} \frac{1}{\sqrt{\alpha_2{}^2 E\left\{\left|S(f)\right|^2\right\}+\sigma_{b2}{}^2(f)}} \tag{3.21}$$

$$\psi_{x1x2}^{SCOT}(f) = \frac{\alpha_1\alpha_2 e^{-i2\pi f\tau_{12}} E\left\{|S(f)|^2\right\}}{\sqrt{\alpha_1^2 E\left\{|S(f)|^2\right\} + \sigma_{b1}^2(f)}} \frac{1}{\sqrt{\alpha_2^2 E\left\{|S(f)|^2\right\} + \sigma_{b2}^2(f)}} \tag{3.22}$$

Dado que

$$\sigma_{bn}^2(f) = E\left\{|B_n(f)|^2\right\} \quad n = 1, 2 \tag{3.23}$$

$$SNR_n(f) = \frac{\alpha_n E\left\{|S(f)|^2\right\}}{E\left\{|B_n(f)|^2\right\}} \quad para\ n = 1, 2 \tag{3.24}$$

Si la relación señal a ruido de ambos micrófonos las suponemos iguales, se obtiene (3.25)

$$\psi_{x1x2}^{SCOT}(f) = \left(\frac{SNR(f)}{1 + SNR(f)}\right) e^{-i2\pi f\tau_{12}} \tag{3.25}$$

Si se considera que SNR es bastante grande para considerar numerador y denominador casi iguales, se obtiene (3.26).

$$\psi_{x1x2}^{SCOT}(f) = e^{-i2\pi f\tau_{12}} \tag{3.26}$$

Por lo que el cálculo de TDE es independiente de la potencia de la señal de la fuente. El método SCOT es en teoría superior al método CCC solo cuando el nivel de ruido es bajo.

3.3.1.c) Transformada de Fase (PHAT)

Esta variación del algoritmo parte de la idea de obtener información de la fase del espectro cruzado ya que es ahí donde se encuentra la información de TDOA en lugar de la amplitud del espectro cruzado. Por lo que si se descarta la amplitud se realiza el cálculo solo con la fase, tal como se describe a continuación.

Para esto la función de pesos en la frecuencia $\phi(f)$ se hace igual de acuerdo a (3.27)

$$\phi(f) = \frac{1}{|S_{x1x2}(f)|} \tag{3.27}$$

Para este caso el espectro cruzado se considera como está dado por (3.28)

$$\psi(f) = e^{-i2\pi f\tau_{12}} \tag{3.28}$$

El cual depende del TDOA τ_{12}, si se sustituye (3.27) y (3.28) en (3.10) se obtiene (3.29) que como puede observarse es una expresión semejante a la transformada de Fourier

$$r_{\mathrm{x1x2}}{}^{PHAT}(\tau) = \int_{-\infty}^{\infty} e^{-i2\pi f(\tau-\tau_{12})} df \tag{3.29}$$

Que es igual a (3.30)

$$r_{\mathrm{x1x2}}{}^{PHAT}(\tau) = \begin{cases} \infty & \tau = \tau_{12} \\ 0 & otros \quad casos \end{cases} \tag{3.30}$$

Este método en general tiene un mejor funcionamiento a comparación del CCC y SCOT apara obtener el TDE para una fuente de voz.

Para el cálculo de GCC-PHAT la aproximación más común es la siguiente, considerando dos señales digitales $x_1(n)$ y $x_2(n)$ adquiridas por los micrófonos, GCC-PHAT está definida por la ecuación (3.31).

$$G\mathrm{CC}\text{-}\mathrm{PHAT}(\mathrm{d}) = IFFT\left\{ \frac{X_1 \cdot X_2{}^{*}}{|X_1||X_2|} \right\} \tag{3.31}$$

Donde d es un intervalo de tiempo sujeto a $|\mathrm{d}| < \tau_{\max}$, mientras X_1 y X_2 son las transformadas discretas de Fourier de x_1 y x_2 respectivamente. La distancia entre micrófonos determina el máximo retraso de tiempo τ_{max}. Se ha demostrado que en condiciones ideales, GCC-PHAT presenta un máximo correspondiente al TDOA. Por otro lado la reverberación puede introducir otros valores máximos que pueden producir una estimación errónea de TDOA. En la Figura 3.4 se muestra la señal resultado de la transformada de fase mapeada en ángulo de dirección θ, las señales son ventanas de 128 muestras a las cuales se les aplica la transformada de fase [4, 16].

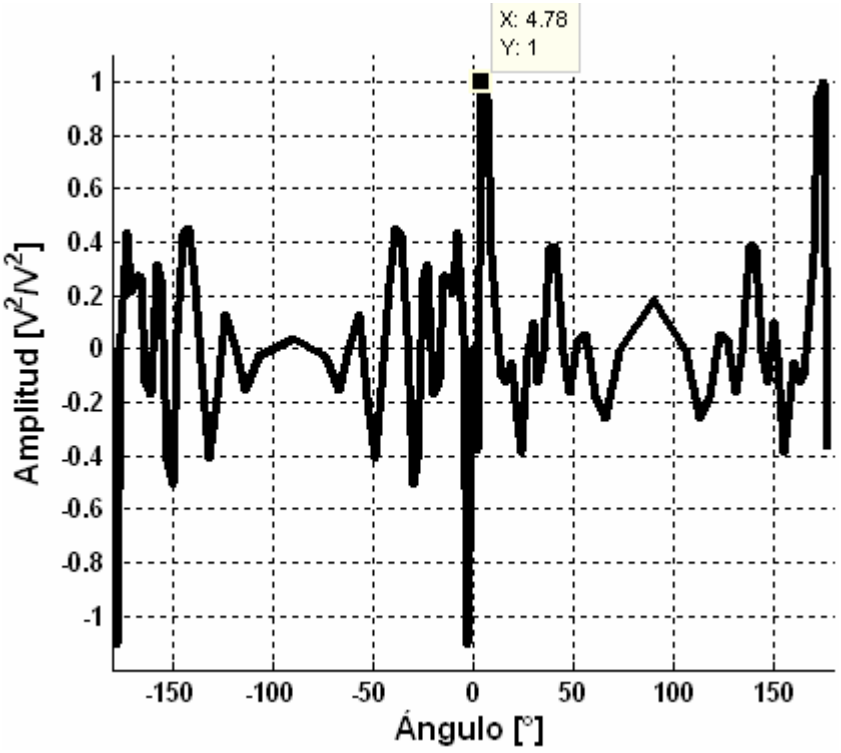


Figura 3.4. Transformada de Fase PHAT de dos señales, mapeada en ángulo de incidencia θ.

Se observa que el valor máximo se encuentra tanto en 4.76° y en 175° respectivamente debido al fenómeno llamado cono de confusión que se verá mas adelante.

3.3.1.d) Algoritmo Transformada de Fase Mejorada (IPHAT)

Los algoritmos PHAT dan buenos resultados bajo condiciones SNR aunque en algunas circunstancias, las interferencias no pueden ser rechazadas tomando en cuenta la ecuación (3.32).

$$G_{12}(\omega) = \alpha_1 \alpha_2 S(\omega) S^*(\omega) e^{-j\omega(\tau_1 - \tau_2)} + N_{1R}(\omega) N^*{}_{2R}(\omega) \tag{3.32}$$

Cuando el SNR es bajo, la relación de $S(\omega)S^*(\omega)e^{-j\omega(\tau_1-\tau_2)}$ en $G_{12}(\omega)$ se hace más pequeño, lo cual hace menos fiable. La existencia de ruido $n_i(t)$ puede causar máximos falsos en $R_{12}(\tau)$ por PHAT.

Para resolver este problema se propuso un PHAT mejorada. El IPHAT se describe a continuación en la ecuación (3.33).

$$\mathrm{R}_{12}(\tau) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{G_{12}(\omega)}{|G_{12}(\omega)|^{\lambda}} e^{j\omega\tau} d\omega \tag{3.33}$$

Donde λ es un factor de alteración acorde a SNR. Una expresión experimental de λ está dada por la ecuación (3.34).

$$\lambda = \begin{cases} \lambda_0 & \sigma < \sigma_0 \\ \dfrac{\lambda_1 - \lambda_0}{\sigma_1 - \sigma_0}(\sigma - \sigma_1) + \lambda_1 & \sigma_0 < \sigma < \sigma_1 \\ \lambda_1 & \sigma > \sigma_1 \end{cases} \tag{3.34}$$

Donde σ representa SNR.

Como consecuencia el factor λ debilita el efecto de blanqueado. En lugar de utilizar solo información de fase en PHAT, también retiene algo de la información amplitud, lo cual reduce las influencias traídas por interferencias utilizando IPHAT. La ecuación (3.33) puede causar más valores máximos en $R_{12}(\tau)$, sin embargo, el valor máximo siempre localizara en tiempo real el retraso, aun en bajas condiciones de SNR [4,16].

3.3.2. Ventajas y Desventajas de los Métodos GCC

Estos métodos tienen las siguientes ventajas:

- En el sentido computacional son eficientes.
- Se puede calcular un TDE en un corto tiempo con lo cual se puede hacer el seguimiento de la fuente de sonido.
- En los estudios realizados han tenido un buen desempeño en ambientes moderadamente ruidosos y no reverberantes, es aquí que surge su mayor desventaja.

Desventajas

- Cuando el ambiente tiene una alta reverberación su funcionamiento decae pudiendo llegar a ser nulo, esto se debe a que los métodos GCC fueron modelados a partir del modelo ideal no reverberante.

3.3.3. Cono de Confusión.

Como se muestra en la Figura 3.6 el sistema tiene dificultades en determinar si el sonido es originado desde la fuente *A* o la fuente *B*.

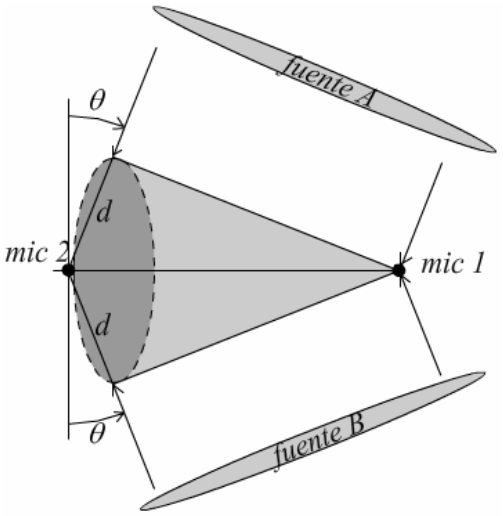


Figura 3.6. Cono de confusión.

Ambas fuentes *A* y *B* se encuentran sobre el cono que envuelve al arreglo de dos micrófonos y que tiene el ángulo *θ,* la detección con los métodos GCC no pueden definir la posición de la fuente ya que las señales obtenidas en los micrófonos resultado del frente de onda tienen el mismo retraso entre ambas[4].

3.4. ALGORITMOS QUE CONSIDERAN LA REVERBERACIÓN

A continuación se verán otros algoritmos los cuales consideran la reverberación del ambiente, el cual es uno de los mayores problemas para obtener TDE, el primero de estos es el algoritmo descomposición de eigenvalores adaptable.

3.4.1. Algoritmo Descomposición de Eigenvalores Adaptable.

Este algoritmo al igual que los anteriores su objetivo es el calculo de TDE, haciendo uso de dos micrófonos, pero este se diferencia en el hecho de que toma en cuenta la reverberación presente en el ambiente, haciendo uso del modelo real reverberante.

Primero identifica las respuestas impulso de los dos canales y mide la diferencia de tiempo entre las dos trayectorias directas reconocidas como el TDOA estimado. Ya que la señal es desconocida, a este método se le considera un método ciego.

Si se considera el modelo real reverberante en ausencia de ruido, se tiene la ecuación (3.35).

$$\mathrm{x}_1(k) * h_2 = s(k) * h_1 h_2 = x_2(k) * h_1 \tag{3.35}$$

De la cual se obtienen las siguientes expresiones vectoriales (3.36) en el tiempo *k*

$$\mathrm{x}^{\mathrm{T}}(k)u = x_1^{\ T}(k)h_2 - x_2^{\ T}(k)h_1 = 0 \tag{3.36}$$

Donde *x(k)* son los vectores de las muestras de señal, *u* la matriz que contiene las respuestas impulso de los dos canales, aunque la respuesta al impulso h_1 se resta a h_2, las cuales se expresan en las ecuaciones (3.37), (3.38) y (3.39)

$$\mathrm{x}(k) = \left[x_1{}^T(k) \quad x_2{}^T(k)\right]^T \tag{3.37}$$

$$u = \left[h_2{}^T \quad -h_1{}^T\right]^T \tag{3.38}$$

$$\mathrm{h}_\mathrm{n} = \left[h_{n,0} \quad h_{n,1} \quad \ldots \quad h_{n,L-1}\right]^T \quad \textit{para n = 1, 2} \tag{3.39}$$

Al multiplicar (3.35) por *x(k)* por el lado izquierdo, se obtiene el valor esperado a su vez, se obtiene la matriz de covarianza de las dos señales de los micrófonos como se muestra en la ecuación (3.40) y (3.41)

$$R_{\mathrm{xx}} = \mathrm{E}\left\{\mathrm{x(k)x}^{\mathrm{T}}\mathrm{(k)}\right\} \tag{3.40}$$

$$R_{\mathrm{xx}}\,\mathrm{u} = 0 \tag{3.41}$$

R_{xx} es la matriz covarianza de las dos señales de los dos micrófonos, el vector *u* consiste de dos respuestas impulso, está en el espacio nulo de *R*. Es decir *u* es el eigenvector de de R_x correspondiente a su eigenvalor cero. Para una matriz se llama rango completo si todas columnas y filas son linealmente independientes y si algunas de sus columnas o filas son dependientes se dice que la matriz es de rango deficiente, considerando esto, si R_{xx} es de rango deficiente para 1[3].

3.5. FUNCIÓN DE TRANSFERENCIA RELACIONADO A LA CABEZA (HRTF)

Esta técnica se ha vuelto muy prometedora ya que permite la localización de sonido en plataforma de robots móviles, es una aproximación como el hombre efectúa la localización de sonido cuyas ventajas son su tolerancia al ruido y la habilidad de localizar sonidos en un espacio 3D. En la Figura 3.7 se muestra el modelo mediante bloques del oído humano, en primera instancia se encuentran las funciones de transferencia que representan la distorsión del sonido al incidir sobre partes del cuerpo tales como los hombros, torso cabeza y pinna o mejor conocido como la oreja, antes de entrar al canal auditivo. Estas funciones de transferencia componen la Head Related Transfer Function HRTF.

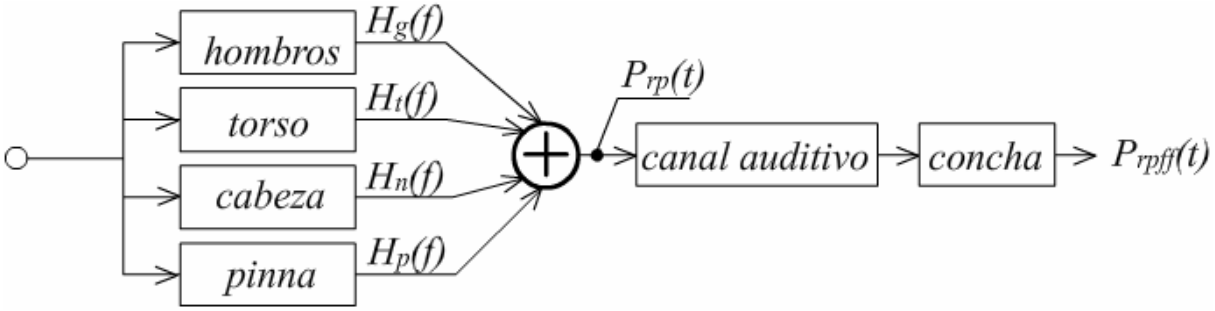


Figura 3.7. Modelo de oído mediante funciones de transferencia.

El uso de HRTF se puede dividir en 3 partes:

- Algoritmos de localización basada en HTRF.
- Reducción de datos HRTF.
- Aplicación de predictores de mejora para la ejecución de localización.

HRTF describe los cambios de ondas de sonido al entrar al canal del oído, debido a la difracción y reflexión del cuerpo humano, la cabeza, brazos, torso y oídos. En aplicaciones de campo lejano, estos pueden ser considerados como funciones de dos variables espaciales (elevación y azimuth) y frecuencias.

HRTF puede considerarse como varios filtros dependientes de la dirección. Como las propiedades difracción y reflexión del cuerpo humano son diferentes para cada dirección. Las características geométricas del cuerpo difieren de persona a persona, los HRTF son únicos por cada individuo [16]. En la Figura 3.8 se muestra el modelo simple de HRTF.

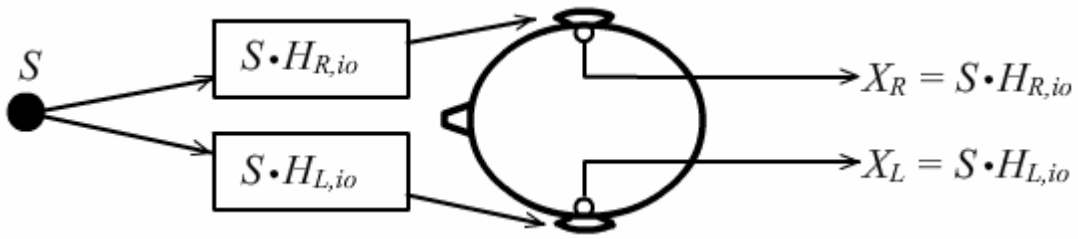


Figura 3.8. Modelo simple de HRTF.

Uno de los problemas de estos métodos es debido a que el cuerpo refracta y refleja la señal original, y las señales que se obtienen, izquierda y derecha son alteradas, el cual el sistema ha aprendido a asociar con una dirección en específico, para un cuerpo distinto se tienen distintas reflexiones y refracciones de las señales.

HRTF es costoso computacionalmente por lo que es mejor reducir la región de búsqueda para fuentes de sonido a una región de interés (ROI). Dado en conjunto HRTF, es necesario observar la presencia de cada HRTF en la señal percibida individualmente.

Algoritmos de localización HRTF

- Filtrado igualador aproximado (matched filtering).

- Cancelación de fuente aproximado.

- Señal referencia aproximado.

- Convolución cruzada aproximada.

Estos algoritmos regresan la posición de la fuente de sonido utilizando las señales del oído grabadas y almacenadas en una base de datos HRTF.

3.5.1. Filtrado Igualador Aproximado (matched filtering)

En la Figura 3.9 se muestra cuando la señal desconocida *S* emitida de una fuente es filtrada por el correspondiente HRTF (izquierdo, derecho) antes de la captura $H_{R,io}$, $H_{L,io}$.

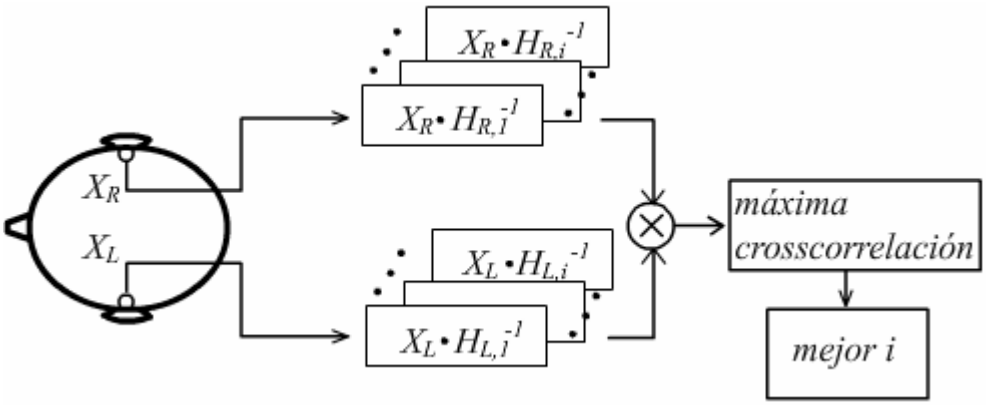


Figura 3.9. Modelo de aproximación filtrado igualador.

La idea es identificar un par de HRTF correspondiente a la posición de emisión de la fuente, tal que la correlación entre los micrófonos izquierdo y derecho es maximizada [16].

Este algoritmo busca los inversos de HRTFs de modo que al filtrar las señales $X_{L,io}$ y $X_{R,io}$ se obtengan las señales $S_{R,i}$ y $S_{L,i}$ es decir, la señal mono aural *S*, tal como puede verse en las ecuaciones (3.42), (3.43) y (3.44).

$$S_{L,i} = H_{L,i}^{-1} \cdot X_L \tag{3.42}$$

$$S_{L,i} = H_{R,i}^{-1} \cdot X_R \tag{3.43}$$

$$S_{L,i} = S_{R,i} \tag{3.44}$$

La fuente se puede localizar por maximización de la correlación cruzada entre $S_{R,i}$ y $S_{L,i}$ como se observa en la ecuación (3.45)

$$\underset{\mathrm{i}}{\operatorname{argmax}} \left\{ S_{R,i} \oplus S_{L,i} \right\} \tag{3.45}$$

Donde el operador $\oplus$ es la correlación cruzada

Algunos puntos importantes al respecto, la inversión de HRTF puede ser problemática por inestabilidad, es por ello que surgió un método para obtener HRTF inversa estable llamado factorización más interno- más externo, que convierte una inversa inestable en una anticausal limitada (finita) inversa.

3.5.2. Algoritmo Cancelador de Fuente

Es una extensión del anterior, en el cual se replantea la correlación cruzada entre los pares $\frac{\mathrm{X_L}}{\mathrm{X_R}}$ y $\frac{\mathrm{H_{L,i}}}{\mathrm{H_{R,i}}}$, como se puede observar son la relación entre las señales filtradas, la izquierda entre la derecha, y la relación de las HRTF izquierda y derecha, de ahí que es una mejora con respecto al anterior ya que no es necesario calcular inversas y pueden ser precalculadas y almacenadas en memoria [16]. Tal como se muestra en (3.46).

$$\frac{\mathrm{X_L}}{\mathrm{X_R}} = \frac{\mathrm{S_{L,i}} H_{L,io}}{\mathrm{X_{R,i}} H_{R,io}} \approx \frac{\mathrm{H_{L,i}}}{\mathrm{H_{R,i}}} \tag{3.46}$$

Se realiza la correlación entre ambas y se obtiene el valor máximo de acuerdo a la expresión (3.47)

$$\underset{\mathrm{i}}{\operatorname{argmax}} \left\{ \frac{\mathrm{X_L}}{\mathrm{X_R}} \oplus \frac{\mathrm{H_{L,i}}}{\mathrm{H_{R,i}}} \right\} \tag{3.47}$$

3.5.3. Aproximación en Base a Convolución

Para evitar problemas de estabilidad, este algoritmo explota la propiedad de asociatividad del operador convolución. La Figura 3.10 ilustra la localización por convolución de una sola fuente. Las observaciones $S_{R,i}$ y $S_{L,i}$ son filtradas con un par de HRTFs contralaterales. Las observaciones filtradas tienden a ser idénticas en la posición correcta de la fuente, para el caso ideal se considera que (3.48) a (3.52) son ciertas[16].

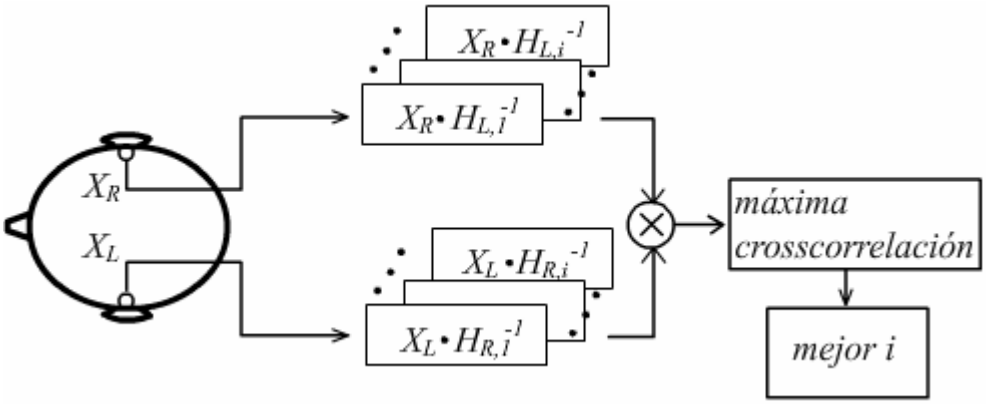


Figura *3.10. Modelo de aproximación en base a convolución.*

$$S_{L,i} = H_{R,i} \cdot X_L \tag{3.48}$$

$$S_{L,i} = H_{R,i} \cdot X_{L,io} \cdot S \tag{3.49}$$

$$S_{L,i} = H_{L,i} \cdot X_{R,io} \cdot S \tag{3.50}$$

$$S_{L,i} = H_{L,i} \cdot X_R \tag{3.51}$$

$$S_{L,i} = S_{R,i} \tag{3.52}$$

Similar al filtrado igualador aproximado, la fuente puede ser localizada en el caso real por resolución del siguiente problema

$$\underset{i}{\text{argmax}} \left\{ S_{R,i} \oplus S_{L,i} \right\} \tag{3.53}$$

De acuerdo a los experimentos se obtienen mejores resultados con la aproximación en base a la convolución.

3.5.4. Señal de Referencia Aproximado

Como se observa en la Figura 3.11, este tipo de aproximación utiliza 4 micrófonos, dos para las señales filtradas HRTF, X_R, X_L y dos fuera del canal del oído para las señales originales $X_{R,out}$, $X_{L,out.}$ Los algoritmos previos utilizan 2 micrófonos, cada uno recibe las señales de sonido HRTF filtradas mono aurales [16].

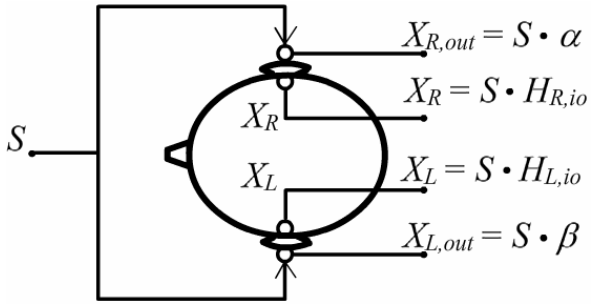


Figura 3.11. Modelo aproximación señal de referencia.

Las cuatro señales capturadas ahora son (3.54), (3.55), (3.56) y (3.57)

$$X_L = S \cdot H_L \tag{3.54}$$

$$X_R = S \cdot H_R \tag{3.55}$$

$$X_{L,out} = S \cdot \alpha \tag{3.56}$$

$$X_{R,out} = S \cdot \beta \tag{3.57}$$

Donde α y β representan retrasos en el tiempo y elementos de atenuación que ocurren debido a las sombras de la cabeza. De estas señales, tres relaciones se calculan $\frac{X_L}{X_{L,out}}$ y $\frac{H_R}{H_{R,out}}$ son las HRTF derecha e izquierda y $\frac{X_L}{X_R}$ es la relación entre la HRTF derecha e izquierda. Las 3 relaciones correlacionadas con las HRTFs respectivas. Los coeficientes correlación cruzada se suman, y el par HRTF produce la suma máxima.

$$\underset{i}{\operatorname{argmax}} \left\{ \left(\frac{X_L}{X_{L,out}} \oplus H_{L,i} \right) + \left(\frac{X_L}{X_R} \oplus \frac{H_{L,i}}{H_{R,i}} \right) + \left(\frac{X_R}{X_{R,out}} \oplus H_{R,i} \right) \right\} \tag{3.58}$$

La ecuación (3.58) define la dirección incidente. La ventaja de este sistema es que HRTFs pueden calcularse directamente reteniendo las señales de sonido sin distorsión $X_{R,out}$, $X_{L,out}$. Por lo que el filtro dependiente de la dirección puede cambiar el espectro incidente sin hacer caso a la información contenida. Aunque al utilizar cuatro micrófonos cambia el concepto de localización biaural.

3.6. MÉTODO SELECCIÓN DE REGIÓN Y GCC SUMADO.

Los métodos basados en TDOA son utilizados asumiendo que no hay objetos interfiriendo entre el par de micrófonos. La dirección de la fuente de sonido es estimada de un tiempo de retraso específico entre los dos micrófonos. Las funciones de mapeo entre TDOA y dirección de

la fuente son representadas por la ecuación y en condiciones de campo libre. Recordando la ecuación (3.1) cuando se utilizan tres o más micrófonos para localizar la fuente en una posición arbitraria a diferencia de utilizar solo dos micrófonos, la dirección de la fuente es estimada por cada par de micrófonos la cual puede ser diferente relativo a cada par. Un arreglo propuesto es el que se muestra en la Figura 3.12 conformado por tres micrófonos colocados sobre los vértices de un triángulo equilátero.

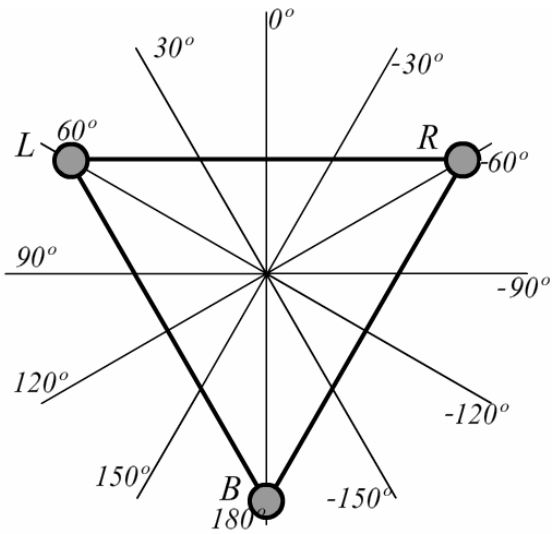


Figura 3.12. Arreglo de tres micrófonos, micrófono derecho R, micrófono izquierdo L, micrófono trasero B.

El método de selección de región consiste en dividir el espacio. En el caso del arreglo de tres micrófonos se divide en tres el arreglo *L – R* lo hace de -30° a 30° y de 150° a -150°, el arreglo *B – R* lo hace de 90° a 150° y de -30° a -90° y el arreglo *L – B* de 30° a 90° y de -90° a -150°. De cada par de micrófonos se realiza la localización de la fuente, tomando el arreglo *L – R* como base, el arreglo *B – L* está girado unos -120° y el arreglo *R – B* está girado unos 120°, así que las funciones correlación cruzada se giran -120° y 120°, en la Figura 3.13 se observan las funciones correlación cruzada de cada par de micrófonos, en la parte de arriba del arreglo *L – R*, en medio el arreglo *R – B* y en la parte de abajo el arreglo *L – B* [16,18].

Como ejemplo considerando que sobre el arreglo de la Figura 3.12 incide una señal de la fuente que se encuentra a cero grados con respecto al sistema coordenado, las señales recibidas se correlacionan y se mapean acorde al sistema coordenado de -180° a 180° como se muestra en la Figura 3.13, correspondiendo cada correlación cruzada a un sub arreglo *L-R*, *R-B* y *B-L*.

En este caso se le llama mapeo al cambio de ejes de tiempo a dirección angular, en los que se encuentra la función correlación cruzada. Se aprecia que para las tres correlaciones cruzadas existen más de un valor máximo, cada máximo muy cercano a 0°.

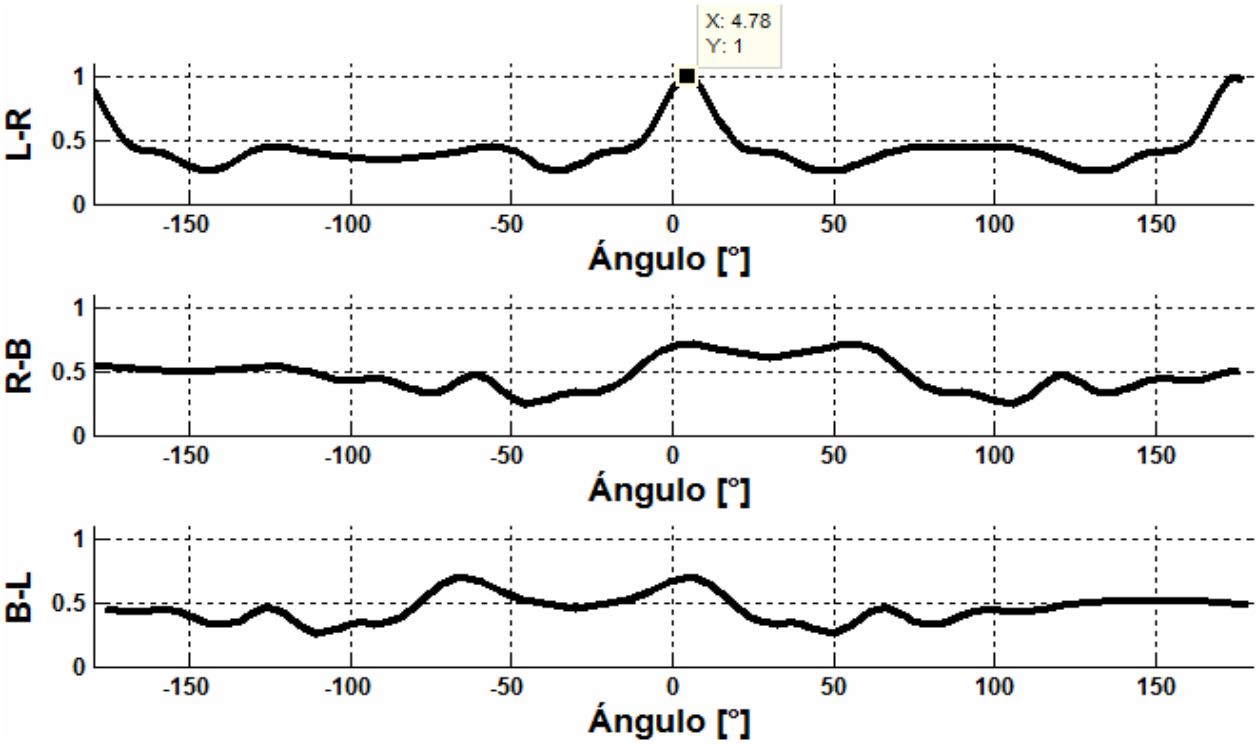


Figura 3.13. Señales correlación cruzada de cada par de micrófonos del arreglo.

Ya con las funciones correlación cruzada se superponen y se observa donde se encuentra el valor máximo de las tres como en la Figura 3.14.

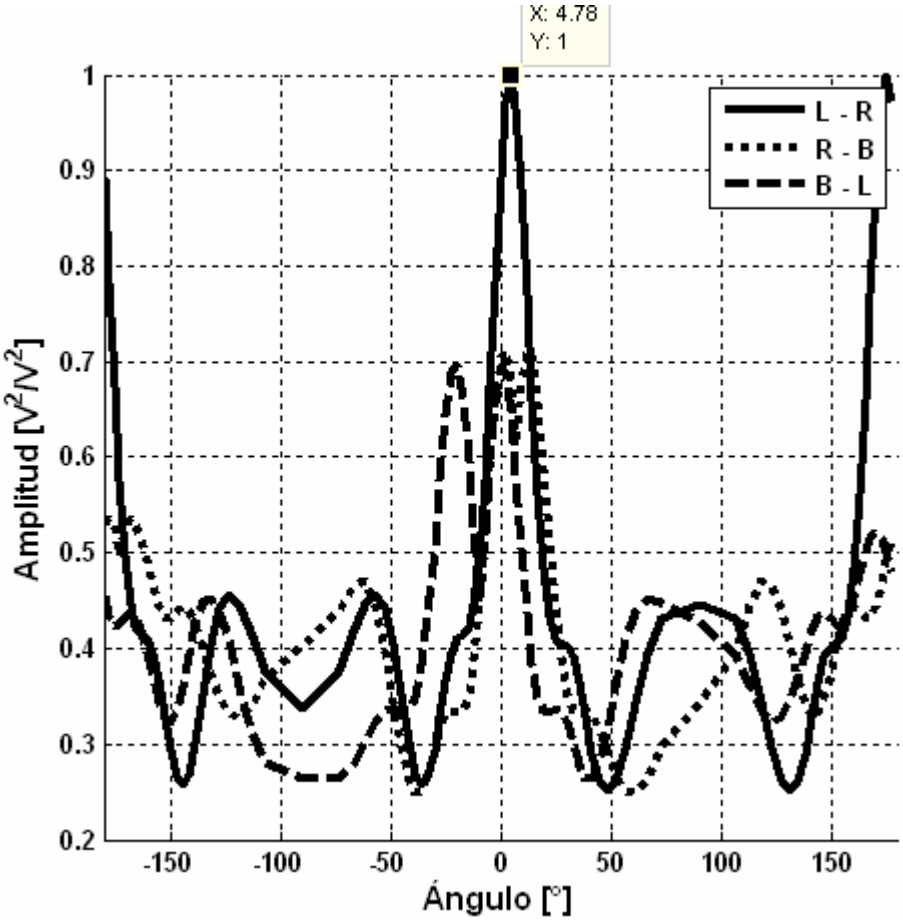


Figura 3.14. Sobreposición de las tres funciones correlación cruzada.

Como se puede observar la estimación de la localización de la fuente con el método selección de región se considera como el valor donde se encuentra el valor máximo de las tres funciones correlación cruzada, en este caso el valor máximo se encuentra en 4.78°, por lo que existe un error de 4.78°.

A partir de este método surge el método GCC sumado, en el cual se utiliza el mismo arreglo, y las mismas funciones correlación cruzada mapeadas de la misma forma, aunque en este caso en lugar de sólo sobreponer las funciones, éstas se suman, como se observa en la Figura 3.15.

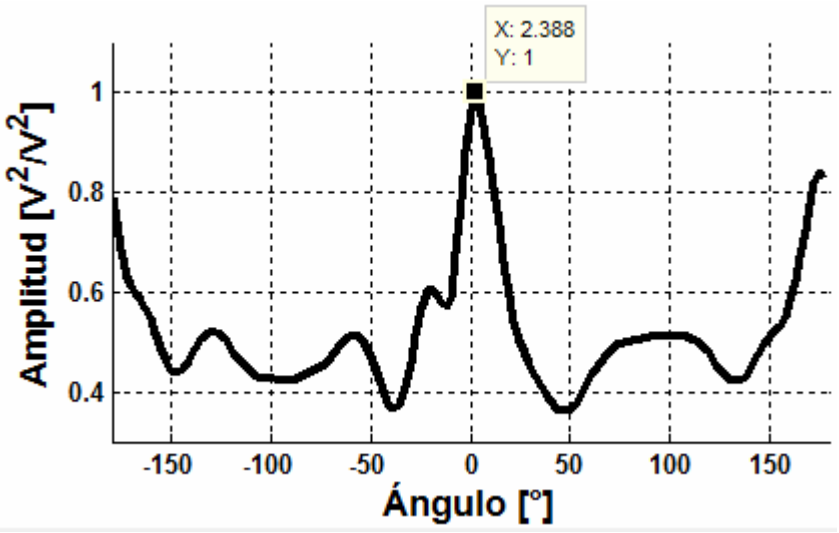


Figura 3.15. Señal suma de las tres señales correlación cruzada, el máximo se encuentra a 2.338°.

Al sumar las tres señales correlación cruzada, se obtiene una función con el valor estimado de la fuente a 2.388°, por lo que el error disminuye de 4.78° a 2.388°, recordando que la fuente se encuentra a 0°.

Debido a imprecisiones en el conocimiento de parámetros del sistema y suponer un modelo no realista como una fuente puntual y campo libre perfecto. Los métodos convencionales estiman la dirección de la fuente para utilizar la información únicamente sobre la TDOA de la correlación cruzada de cada par de micrófonos. Así, la dirección de la fuente actual es determinada por las direcciones de fuente estimadas de cada par de micrófonos y el criterio de error específico.

De cualquier forma, los métodos GCC sumados utilizan la información total de la función correlación cruzada, la cual es una función en el dominio del tiempo. Las funciones correlación cruzada de cada micrófono en el dominio del tiempo son transformadas a la nueva función en dominio espacial mediante funciones de mapeo específicos. A partir de las nuevas funciones son representadas por el mismo dominio espacial, las funciones mapeadas son sumadas dentro de una única función y entonces una fuente de sonido es determinada al hallar el máximo de la función única sumada. Esta aproximación es el llamado GCC sumado [6,16].

3.7. LOCALIZACIÓN 3D CON MÉTODO GCC SUMADO

Con el arreglo mostrado en la Figura 3.12 se puede realizar la localización 3D para ello los micrófonos no deben estar colocados en el plano central ya que es necesario discriminar la confusión entre arriba y abajo. Si los micrófonos están en el plano central es imposible localizar la fuente en espacio 3D únicamente.

Si las mediciones HRTF están disponibles, las funciones mapeo pueden ser obtenidas de las HRTF medidas. Las funciones mapeo son calculadas por la solución analítica de la función de transferencia relacionada a “cabeza esférica” SHRTF. Estas funciones de mapeo en la elevación

simétrica son diferentes a aquellas de condiciones de campo libre. Estas diferencias hacen posible la localización de la fuente en el espacio 3D. Ambos ángulos de elevación y azimuth de la fuente de sonido son estimadas simultáneamente por el método GCC sumado con las funciones mapeo totales en espacio 3D. Las funciones correlación cruzada son mapeadas para direcciones de fuente sobre ángulos azimuth y elevación. Cuando una fuente es localizada en 120º azimuth y -90º elevación, las funciones correlación cruzada de cada par de micrófonos son mapeadas para dirección fuente por funciones mapeo en espacio 3D.

La resolución de funciones mapeo sobre la elevación es 10 grados. Después de ser mapeadas a la dirección fuente, allí aparecen líneas de corte circular con la misma magnitud debido al cono de confusión. El cono de confusión se aprecia como un círculo perfecto en condiciones de campo libre [9].

En el proyecto se utiliza el método de GCC sumado y transformada PHAT para realizar la localización de una fuente de voz ya que pueden obtener un estimado de dirección de 360° con tres micrófonos.

3.8. SÍNTESIS

En el capitulo se tratan algunos métodos de localización de una fuente de sonido, en especial para arreglo de dos a tres micrófonos en la dirección de azimuth. Primero es necesario conocer como se propaga en un medio el sonido proveniente de una fuente, hay dos modelos que explican la propagación, el modelo ideal de campo libre y el modelo real reverberante, el primero considera que no existen objetos en el espacio donde se propaga la onda que produzcan reflexión o reverberación, y el segundo toma en consideración el efecto de la reverberación la cual modela como una función de transferencia del medio que convoluciona a la señal de la fuente de onda.

Dependiendo del modelo considerado, se presentan distintos métodos para la localización, se presentan aquellos que consideran dos micrófonos, tales como los métodos GCC que consideran el modelo ideal de campo libre para hacer un estimado del TDOA o como el Algoritmo Descomposición de Eigenvalores Adaptable, HRTF que consideran el modelo real reverberante.

Después se presenta el método de división de Selección de Región y GCC Sumado que utilizan el GCC, los cuales realizan la localización mediante un arreglo de tres micrófonos ubicados en los vértices de un triángulo equilátero el cual se divide en sub arreglos de dos micrófonos dividiendo el espacio en tres. Se hace esta división ya que en ciertas áreas un sub arreglo puede realizar mejor la localización que otro que cubre una región distinta.

Al final, se presentó cómo se puede realizar la localización en 3D, es decir, realizar la localización tanto de azimuth como de elevación, mediante el método de GCC Sumado y se requiere tener las HRTF del sistema.

- Página en blanco intencionalmente -

4

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

El diseño del sistema consiste en un arreglo de tres micrófonos dispuestos sobre un plano, las señales obtenidas son enviadas a un circuito acondicionador de señal para que sean filtradas y amplificadas, a su vez son adquiridas por el periférico convertidor analógico digital de la tarjeta de desarrollo DSK TMS320F2812 para su digitalización, como se muestra en la Figura 4.1. Después los datos son procesados por la tarjeta de desarrollo DSK TMS320C6416T de Texas Instruments con la cual se obtiene un estimado de la dirección de la fuente acústica a partir de las señales recibidas en los micrófonos.

En este capítulo se explican cada parte del sistema así como los criterios tomados en cuenta para su diseño. En la Figura 4.1 se muestra el esquema general del sistema.

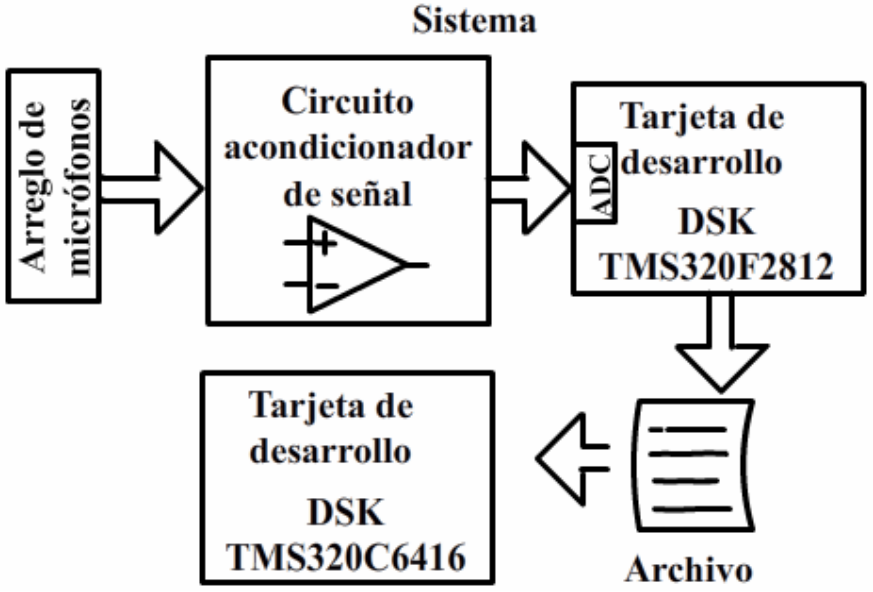


Figura 4.1. Esquema general del sistema.

4.1. CARACTERÍSTICAS DSP TMS320C6416 Y TARJETA DE DESARROLLO DSK TMS320C6416T

Para el procesamiento de las señales se eligió en DSP TMS320C6416 de la familia C6000 a continuación se describen algunas de sus características. Los dispositivos C6000 ejecutan hasta 8 instrucciones de 32 bits por ciclo de reloj. El núcleo del CPU de C64x consiste de 64 registros de propósito general de 32 bits y 8 unidades funcionales. En la Figura 4.2 se presenta un esquema de las unidades del DSP. Estas unidades funcionales contienen:

- 2 Multiplicadores.
- 6 Alus.
- 1 MByte interno de RAM.
- Interface de expansión HPI.
- Emulador embebido JTAG.
- Opera a 1 GHz en aritmética punto fijo.

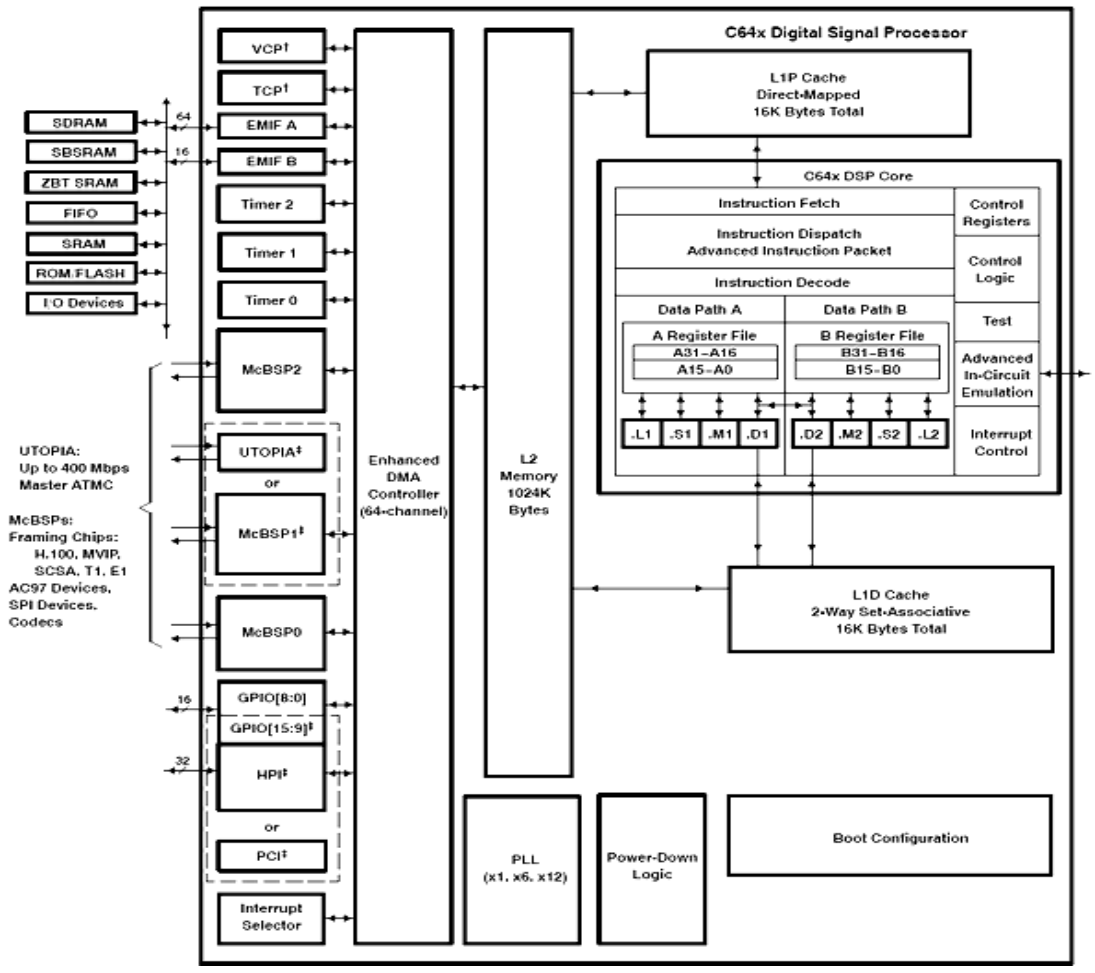


Figura 4.2. Diagrama CPU C64x.

La tarjeta de desarrollo DSK TMS320C6416T es una plataforma que permite evaluar y desarrollar aplicaciones para la familia TI DSK C64xx. El DSK también sirve como una referencia de diseño de hardware con DSP TMS320C6416T.

La tarjeta DSK viene con los siguientes dispositivos que se ajustan a una amplia variedad de aplicaciones.

- Un DSP TMS320C6416T de Texas Instruments operando a 1GHz.
- SDRAM externo de 16 MBytes
- Flash externo 512 KBytes, interface 8 bit
- Codec AIC23
- CPLD lógica programable
- 4 leds usuario
- 4 dip switch
- 3 switches de configuración
- Un códec de audio AIC23 estereo.
- Conectores estándar de expansión para utilizar tarjetas hijas.
- Fuente unipolar de +5 V
- Emulación JTAG a través de emulador JTAG con interfaz USB.

En la Figura 4.3 se presenta una fotografía de la tarjeta de desarrollo DSK TMS320C6416T en el cual se depuraron los programas que más adelante se describen.

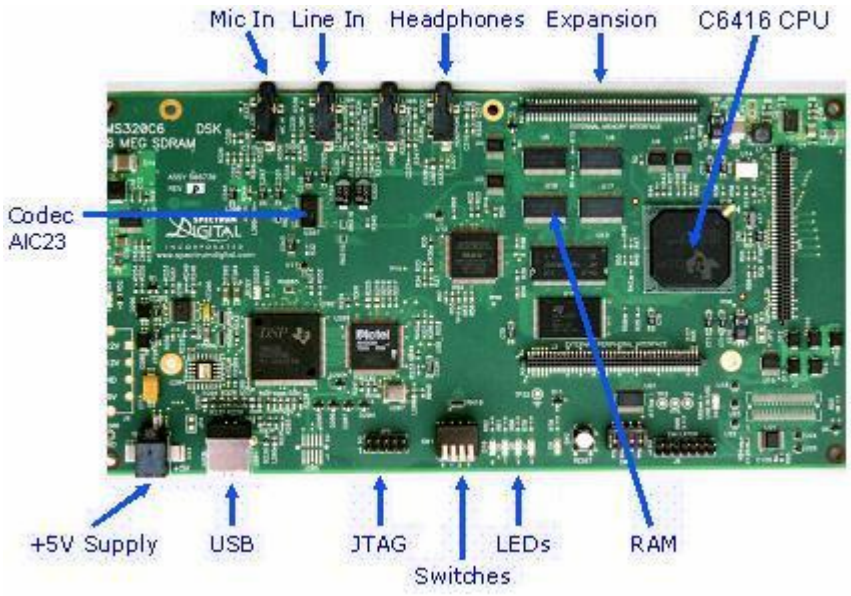


Figura 4.3. Tarjeta de desarrollo DSK TMS320C6416T

4.2. ARREGLO DE MICRÓFONOS

Consiste de un conjunto de tres micrófonos ubicados en un plano y separados una distancia *d*. A continuación se mencionan los criterios tomados en cuenta en el diseño del arreglo,

el tipo de micrófonos que se utilizan, la distancia de separación entre micrófonos y su distribución en el espacio.

4.2.1. Micrófono Electret em-926

Se eligió este micrófono por varias razones las cuales son:

- Bajo costo. Existen en el mercado otros tipos de micrófonos con características superiores pero sus altos costos son una limitante a diferencia del micrófono electret.

- Directividad omnidireccional. Los sensores del arreglo deben ser omnidireccionales, para que las señales recibidas no sean atenuadas dependiendo de la dirección de la cual se reciben.

- Baja potencia. Dado el número de componentes tarjeta de desarrollo, circuito acondicionador, convertidor analógico digital y tarjeta interface, se requiere que la potencia sea mínima, el micrófono electret requiere bajo voltaje y corriente de polarización para su funcionamiento.

- El intervalo de frecuencias útil es de 20 Hz a 16 kHz.

- Las dimensiones del micrófono son lo suficientemente reducidas en comparación de otros micrófonos, con un diámetro de 9.7 mm y 5.2 mm de espesor.

4.2.2. Diseño del arreglo de micrófonos

El arreglo elegido para efectuar la localización es un arreglo planar cuya ubicación de los tres micrófonos se encuentra sobre los vértices de un triángulo equilátero la longitud de cada lado del triángulo es de 0.17 metros.

Aunque en teoría, para efectuar la localización por GCC sumado los elementos se consideran como puntuales, pero en realidad estos tienen dimensiones. En la Figura 4.4 se muestran las dimensiones del micrófono electret em-926.

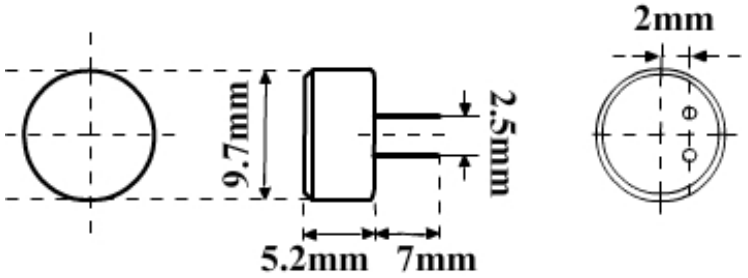


Figura 4.4. Dimensiones micrófono electret em-926.

La distancia se mide a partir del centro de la cara circular de los micrófonos. Los micrófonos se distribuyen sobre los vértices de un triángulo equilátero, ya que cada sub arreglo puede realizar la estimación de una fuente de sonido por si solo para las direcciones de -30° y 30°. Es así, que para aumentar la precisión del sistema se utilizan tres sub arreglos que dividen el

espacio en tres, de esta forma los ángulos para los cuales pierde precisión un sub arreglo, otro sub arreglo puede realizar la localización en esa posición. Tal como se explica en el capítulo tres cuando se describe el método de selección de región y GCC sumado.

Se realizó en un circuito impreso, para utilizar el arreglo de micrófonos que se muestra en la Figura 4.5.

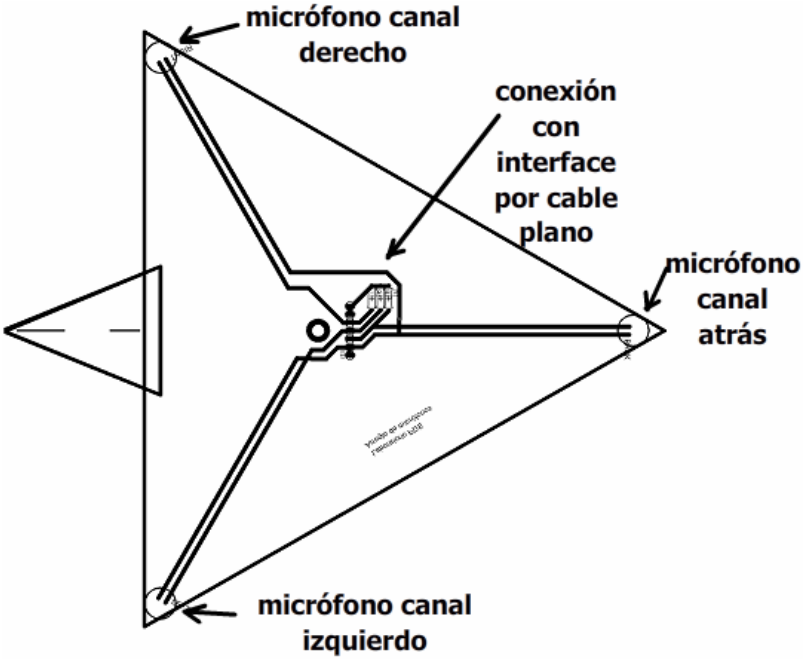


Figura 4.5. Circuito impreso del arreglo de micrófonos.

Mediante un cable plano se conecta el arreglo de micrófonos con el circuito acondicionador de señal, el triángulo a la izquierda del circuito impreso se utiliza para indicar el ángulo del arreglo en un transportador como se observa en el capítulo 5.

4.3. CIRCUITO ACONDICIONADOR DE SEÑAL

El voltaje de la señal de salida de los micrófonos es del orden de milivolts, lo cual no es suficiente para ser detectado por el convertidor analógico digital, el convertidor realiza la conversión de señales en el intervalo de 0 a 3 V, por lo cual es necesario amplificar la señal, además que la señal de entrada esté limitada en banda.

El circuito está diseñado con un ancho de banda de 20 Hz a 16 kHz, se utiliza el circuito amplificador operacional TL074 debido a su de ganancia unitaria ancho de banda BW de 4 MHz, parámetro que limita la ganancia y ancho de banda del circuito. En la Figura 4.6 se muestra el esquema del circuito acondicionador de señal, los voltajes de polarización son de +/- 5 V, la resistencia R_r sirve para polarizar al micrófono la cual alimenta una pequeña corriente de cuyo valor es de 1.5 kΩ, el capacitor C_1 en serie con la resistencia R_1 forman un filtro paso altas de acuerdo a la ecuación (4.1), el cual elimina la componente de directa de la señal proveniente del micrófono, así como atenúa las frecuencias menores la capacitancia es de 2.2 μF y la resistencia de 3.9 kΩ. Con lo que la frecuencia de corte f_L es de 18.5 Hz cercanos a los 20 Hz.

La capacitancia C_2 y R_2 en paralelo de la Figura 4.6 forman un filtro paso bajas que limita en banda a las señales y evitan el fenómeno de aliasing, sus valores son de 15 pF y 680 kΩ que de acuerdo a la ecuación (4.2), la frecuencia de corte f_H es de 15.6 kHz próximos a los 16 kHz.

$$f_L \approx \frac{1}{2\pi R_1 C_1} \tag{4.1}$$

$$f_H \approx \frac{1}{2\pi R_2 C_2} \tag{4.2}$$

La ganancia del amplificador depende de las resistencias R_1 y R_2. Para frecuencias centrales donde se puede despreciar el efecto de los capacitares C_1 y C_2 se aproxima mediante la ecuación (4.3), dado que R_1 es de 3.9 kΩ y R_2 de 680 kΩ la ganancia es de aproximadamente -174.

$$A \approx -\frac{R_2}{R_1} \tag{4.3}$$

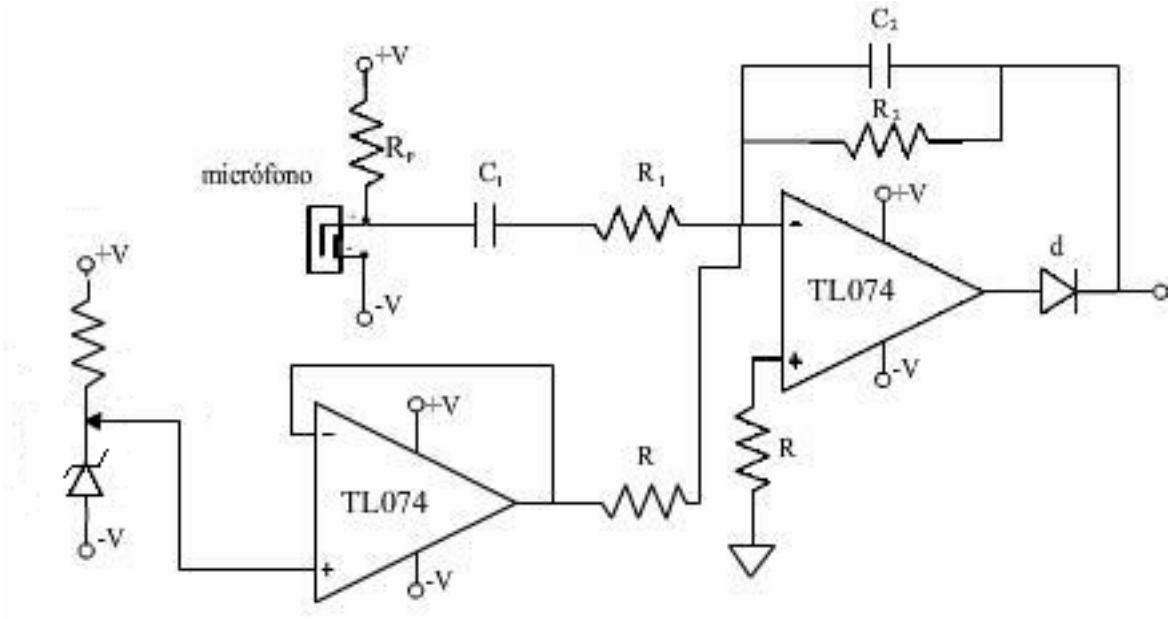


Figura 4.6. Diagrama de circuito acondicionador de señal.

El seguidor de voltaje que se observa se encarga de alimentar un voltaje de DC en este caso de -1.5V que sirve para dar un voltaje de offset a las señales, el cual es necesario para el convertidor analógico digital, el diodo zener es de 3.6 V zener y la resistencia que lo alimenta es de 2.2 kΩ, de esta manera se obtiene -1.5 V que al invertirse dan el voltaje de offset de 1.5 V.

4.4. CONVERTIDOR ANALÓGICO DIGITAL

Dado que la tarjeta de desarrollo DSK TMS320C6416T no cuenta con un convertidor analógico digital con los canales suficientes, se recurrió a capturar las señales con la tarjeta de desarrollo DSK TMS320F2812, que se muestra en la Figura 4.7.

Figura 4.7. Tarjeta de desarrollo DSK TMS320F2812 del cual se obtienen las señales.

Cuyo DSP tiene un convertidor analógico digital ADC interno de 16 canales que funciona a una frecuencia de 20 MHz. En la Figura 4.8 se muestra un esquema del convertidor, este periférico se configura para hacer la captura de señales a una frecuencia de muestreo de 48 kHz y a continuación se explica el porque se eligió esta frecuencia. El programa que configura el DSK TMS320F2812 junto con su periférico ADC se encuentra en el apéndice C.

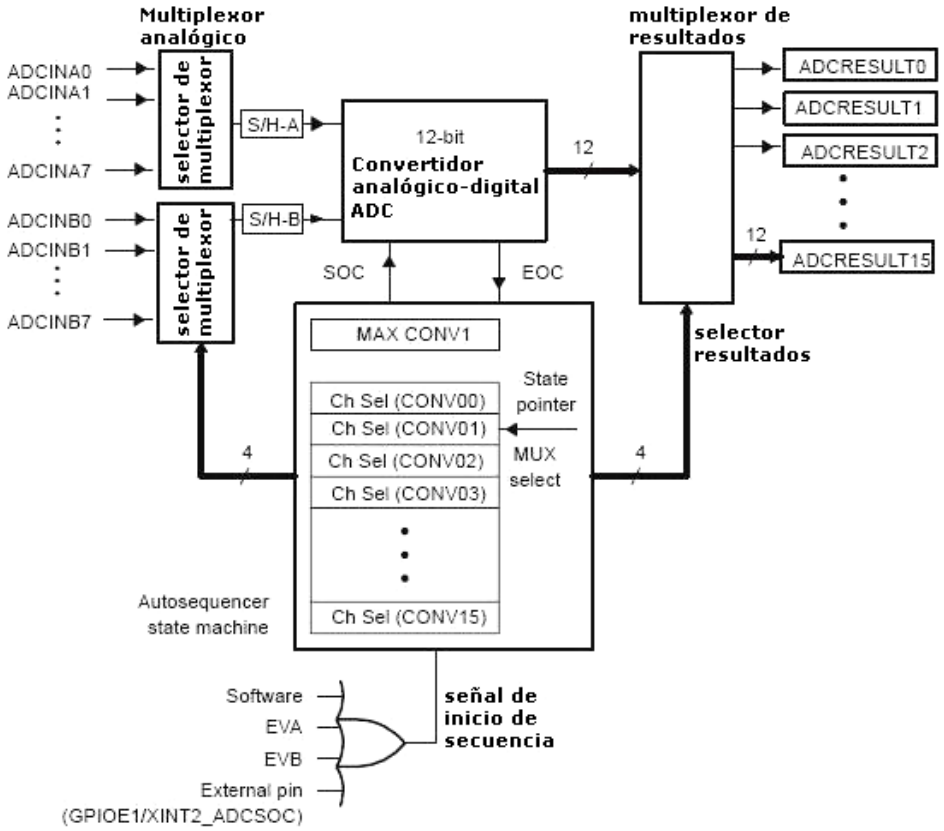


Figura 4.8. Esquema de ADC interno del DSP TMS320F2812 con el que se digitalizan las señales.

Las tres señales adquiridas se guardan en vectores de 2731 muestras cada uno, correspondientes a los canales derecho *R*, izquierdo *L* y atrás *B*. Los datos se almacenan en un archivo con terminación ".dat" con la herramienta de Code Componer que permite guardar memoria en un archivo, en formato hexadecimal.

Los archivos adicionalmente se procesan con una función programada de matlab manArchivos(), incluida también en el apéndice A de manera que los tres vectores de 2731 muestras se ordenan en vectores de 128 muestras y de 256, es decir, los datos se pasan por la ventana de 128 y 256 muestras, y los datos se ordenan para que coincidan con el tipo de dato de la tarjeta.

Las muestras solo se cargan en memoria para su procesamiento de manera que coincidan en las localidades de los vectores correspondientes a los canales derecho R, izquierdo L y atrás B.

4.5. FRECUENCIA DE MUESTREO

La frecuencia de muestreo se eligió de 48 kHz, ya que con esta se obtiene una buena resolución y mapeo de la estimación de dirección de la fuente de sonido. Como se vió en el capitulo tres, cada canal tiene un retraso relativo con otro para un sub arreglo de dos micrófonos, dicho retraso está dado por la ecuación (4.4)

$$\tau = \frac{2r\sin(\theta)}{c} \tag{4.4}$$

Donde *2r* es la distancia de separación, θ es el ángulo entre el frente de onda plano y el sub arreglo, τ es el retraso de tiempo y es una variable continua, para discretizarla se divide por el periodo de muestreo o se multiplica por la frecuencia de muestreo *Fs* dando como resultado (4.5)

$$\tau' = \frac{2r(Fs)\sin(\theta)}{c} \tag{4.5}$$

Con la ecuación (4.5) podemos hacer el mapeo del retraso de la señal con el ángulo correspondiente. Por ejemplo, considerando frecuencias de muestreo de 48000, 22000 y 8000 Hz se obtienen los siguientes datos mostrados en la Tabla 4.1.

	Retrasos		
Ángulo [°]	48 kHz [muestras]	22 kHz [muestras]	8 kHz [muestra]
0	0	0	0
5	2.0917	0.9587	0.3486
10	4.1676	1.9101	0.6946
15	6.2117	2.8470	1.0353
20	8.2085	3.7622	1.3681
25	10.1428	4.6488	1.6905
30	12	5.5	2
35	13.7658	6.3093	2.2943
40	15.4269	7.0707	2.5712
45	16.9706	7.7782	2.8284
50	18.3861	8.4265	3.0642
55	19.6596	9.0107	3.2766
60	20.7846	9.5263	3.4641
65	21.7514	9.9694	3.6252
70	22.5526	10.3366	3.7588
75	23.1822	10.6252	3.8637
80	23.6354	10.8329	3.9392
85	23.9087	10.9581	3.9848
90	24	11	4

Tabla 4.1. Relación muestra retraso en relación al ángulo estimado por sub arreglo.

Se observa que para la frecuencia de 48 kHz, se tiene 24 muestras para representar el retraso correspondiente a la variación de ángulo de incidencia del frente de onda plano sobre el arreglo y además se aprecia que hasta 70° se pueden representar con la suficiente diferencia entre muestras de al menos una muestra. En cambio, para 22 kHz de 40° en adelante se observa una menos diferencia entre muestras, con lo que se presentan confusiones y para 8 kHz en 5° se hace difícil distinguir con 10°. Es por ello que se decidió por los 48 kHz como frecuencia de muestreo ya que se tienen las suficientes muestras para representar los ángulos de incidencia de frente de

onda. Estas muestras corresponden a las muestras centrales de la función correlación cruzada como se observa en el capitulo 3.

4.6. ESTIMACIÓN DE DIRECCIÓN DE FUENTE POR GCC SUMADO

Para el proceso de estimación se han propuesto dos algoritmos, el primero con la correlación cruzada de las señales, y el segundo mediante la transformada *PHAT*, y se pretende comparar los resultados de ambos. Para cada método se realizó un programa distinto en dos proyectos distintos de Code Componer, los programas están escritos en lenguaje C.

4.6.1. Método GCC Sumado

Para programar la correlación cruzada, que es la base del método GCC sumado, se eligió ventanas de 128 muestras, ya que en este caso es el tamaño de ventana mínimo para efectuar la estimación a una frecuencia de muestreo de 48 kHz. Considerando que las muestras provienen de un archivo de datos o del convertidor analógico digital, las muestras con formato entero signadas con Qi de 11. Inicialmente se tienen seis vectores de datos inicialmente de 128 localidades, dos por cada canal. Al principio ningún vector es procesado, y en cada interrupción generada por el timer 0 que controla la frecuencia de muestreo, se va guardando una muestra de cada canal en sus tres primeros vectores correspondientes mediante apuntadores que al guardar el dato incrementan apuntando a la siguiente localidad. Se tiene un contador que inicialmente vale cero, con cada interrupción su valor se incrementa, de esta forma se controla el tamaño de la ventana ya que una vez que alcanza el valor de 128 se reinicia a cero. Luego se cambian los apuntadores para que apunten a la segunda triada de vectores de datos de canal que son los datos a ser llenados en cada interrupción, a la vez la variable “lleno” que inicialmente valía cero se cambia su valor a uno, lo cual sirve para indicar que la primera triada de vectores está lista para ser procesada, todo esto sucede después de los primeros 2.7 ms.

Con la variable lleno igual a uno se inicia el procesamiento de los primeros tres vectores R, L y B mientras los otros tres se están llenando en cada interrupción, primero se envían a la función de C:

void crosscorrelacion(short *ap1, short *ap2, short tam, short sam, int *res);

Los parámetros de la función son los apuntadores **ap1* y **ap2* que apuntan a los vectores de datos a correlacionar, *tam* es la variable que indica el tamaño de los vectores ya que la ventana es de 128 muestras *tam* es igual a la ventana, dado que no se requiere la función correlación cruzada completa de acuerdo a lo mencionado en el capitulo tres, solo se requieren los valores mas centrales de la función, y la variable *sam* indica el valor del número de muestras centrales entre dos menos uno, es decir, dado que se requieren las 49 muestras centrales *sam* es igual a 24. Y el apuntador **res* apunta a un vector de 49 datos en donde se almacena el resultado, es decir, los datos centrales de la función correlación.

Se efectúan las tres correlaciones cruzadas de los tres sub arreglos, es decir, la correlación cruzada de acuerdo a la Tabla 4.2.

Sub arreglo	*Canal 1*	*Canal 2*	*Función correlación cruzada*
1	*Izquierdo – L*	*Derecho – R*	*LR*
2	*Derecho – R*	*Trasero – B*	*RB*
3	*Trasero – B*	*Izquierdo – L*	*BL*

Tabla 4.2. Correlación cruzada de canales.

Una vez obtenidas las funciones correlación cruzada se pueden mapear los ángulos relativos a cada sub arreglo, dado que son 49 datos, éstos se pueden mapear al intervalo angular de -90° a 90° con respecto a la normal de cada sub arreglo. Como se muestra en la Figura 4.9, considerando que las funciones son simétricas debido al cono de confusión visto en el capítulo 3, se tiene que del intervalo de -180° a -90°, es simétrico con el intervalo de -90° a 0°, y a su vez el intervalo 0° a 90° es simétrico al intervalo de 90° a 180°. Tal como se muestra en la Figura 4.9.

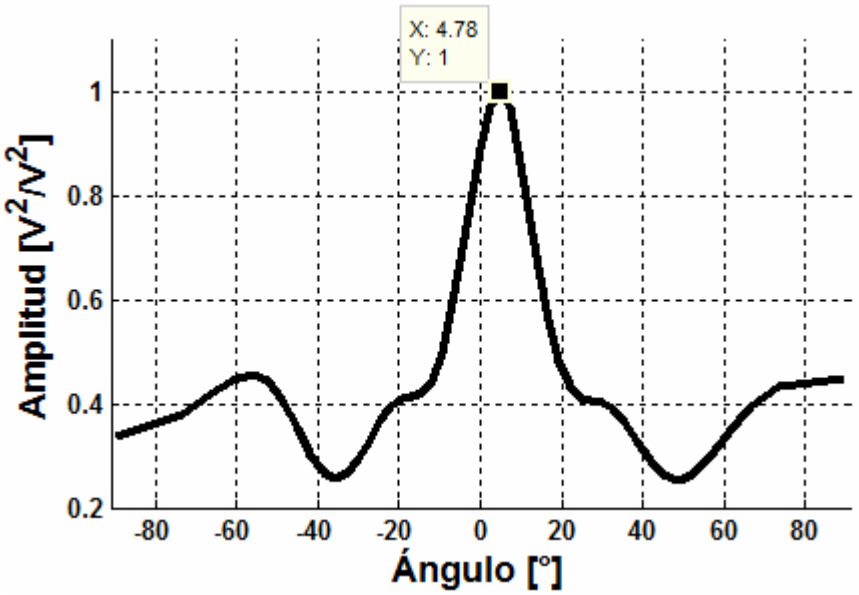


Figura 4.9. Mapeo de la función correlación cruzada al intervalo angular -90° a 90° relativo a cada sub arreglo.

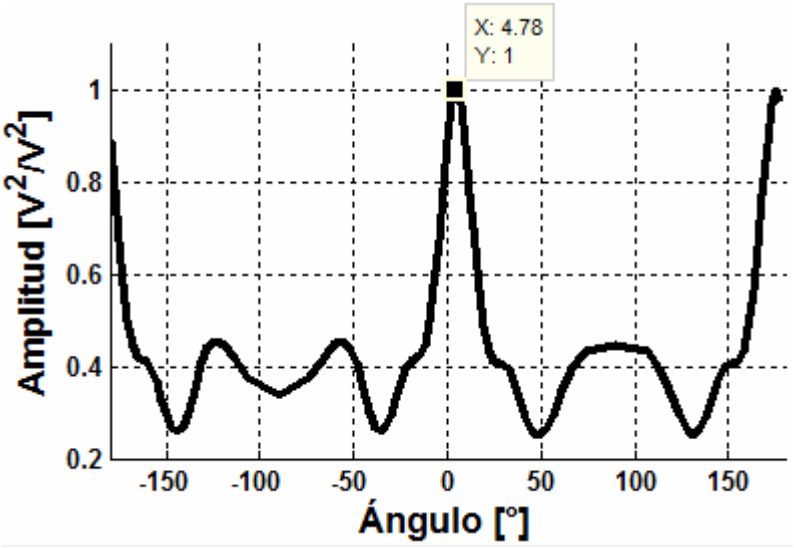


Figura 4.10. Mapeo de la función correlación cruzada al intervalo angular -180° a 180°.

Entonces se tienen los vectores LR, RB, BL resultado de la función correlación cruzada de acuerdo a sus correspondientes sub arreglos, como se muestra en la Figura 4.13, cada muestra es corresponde al ángulo relativo de cada sub arreglo, es decir, el ángulo de cada sub arreglo se mide a partir de la normal de cada uno, es por ello que se requiere que los tres sub arreglos tengan el mismo sistema coordenado, es por ello que se reordenan los datos, dando un corrimiento angular a los datos del vector correspondiente de RB de 120° y al vector BL un corrimiento de -120°. Tal como se muestra en la Figura 4.11, el mapeo de las tres funciones están referenciadas al mismo sistema coordenado.

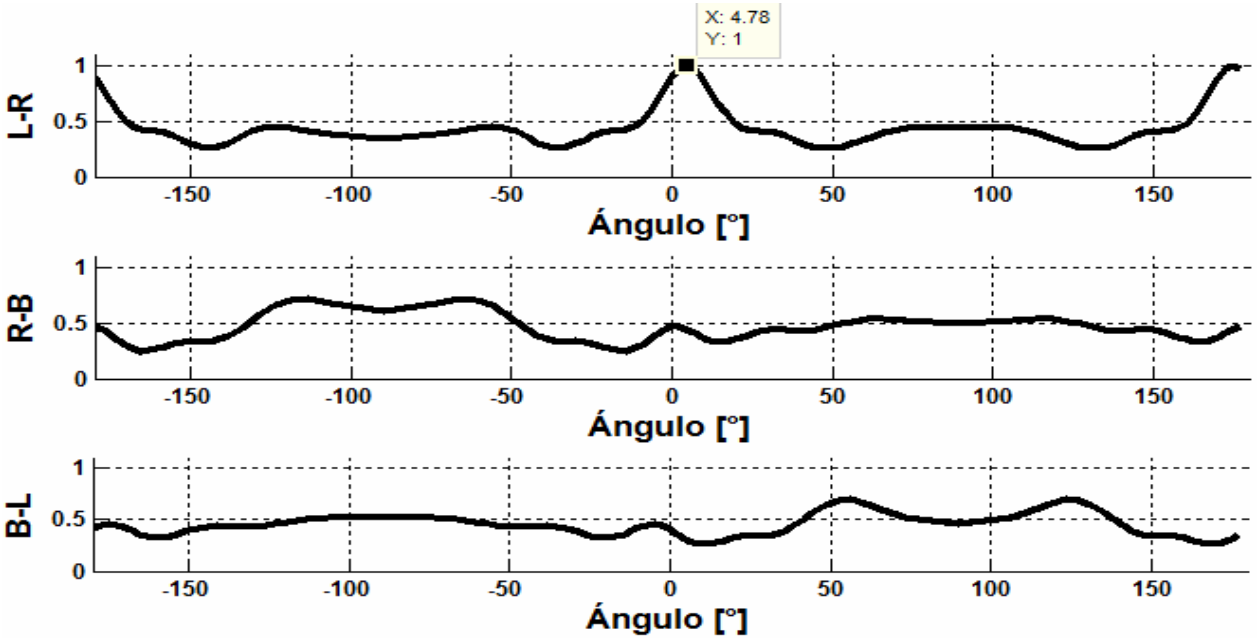


Figura 4.11. Sistema coordenado relativo a cada sub arreglo.

Se utiliza la normal al sub arreglo LR como referencia, al cual los sub arreglos RB y BL deben de estar orientados. Los mapeos angulares están dados por la función reordena(), la cual obtiene la función que mapea de -180° a 180° a partir de la original que solo es de -90° a 90°:

void reordena(int *v1, int *v2, int *v3, int *d1, int *d2, int *d3);

La función recibe los apuntadores *v1, *v2 y *v3, que apuntan a los vectores de 49 muestras donde se guarda el resultado de la función crosscorrelacion(), además los apuntadores *d1, *d2, *d3, apuntan a los vectores donde se guardan los nuevos vectores de 96 muestras, mapeados de -180° a 180°. En el código se ingresa la instrucción:

reordena(LR, RB, BL, LR1, RB1, BL1);

Donde LR, RB y BL, son los vectores resultado de correlación cruzada de 49 muestras cada uno y LR1, RB1 y BL1 son los vectores de 96 muestras correspondientes al mapeo de -180° a 180° de cada sub arreglo con el mismo sistema coordenado. Como se muestra en la Figura 4.12.

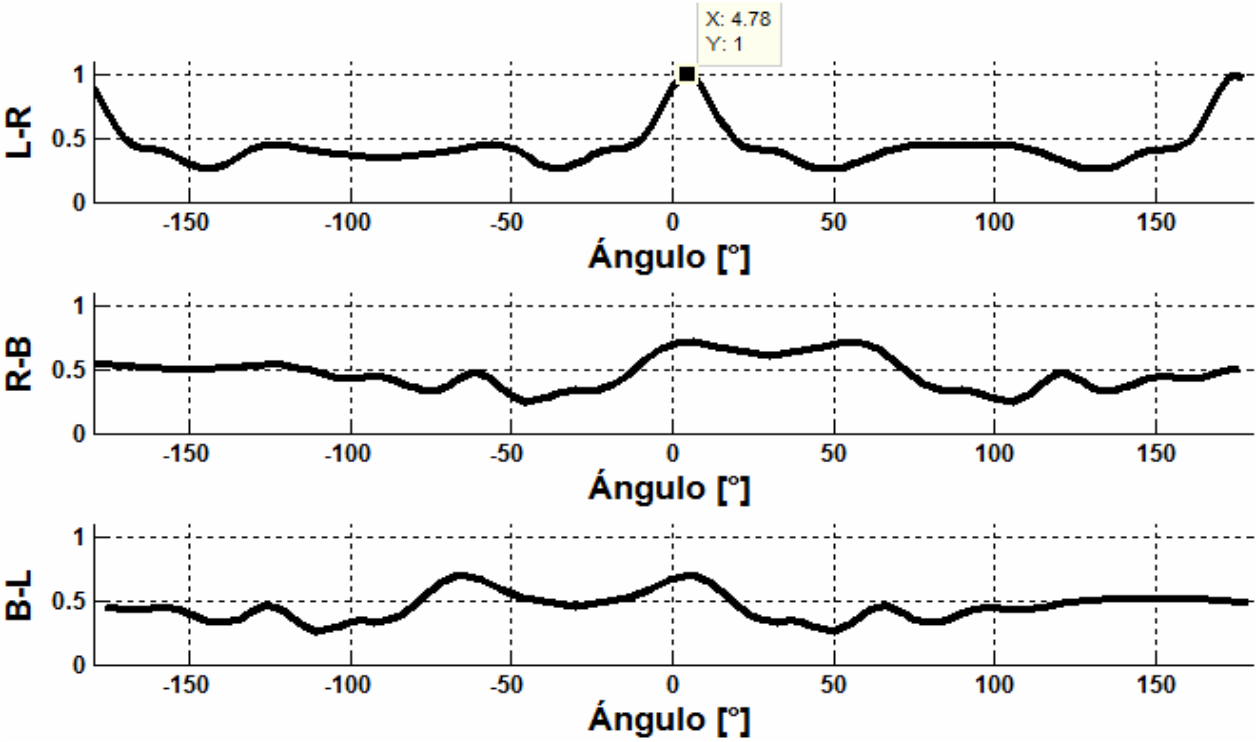


Figura 4.12. Señales referenciadas al mismo sistema coordenado.

En la siguiente parte del código se suman los tres vectores LR1, RB1 y BL1 y el resultado se guarda en el mismo LR1, como se muestra en la Figura 4.13, con la función suma(), pero la suma no se hace directamente sino que se toma en cuenta el desplazamiento angular entre los tres vectores:

void suma(int *LR1, int *RB1, int *BL1);

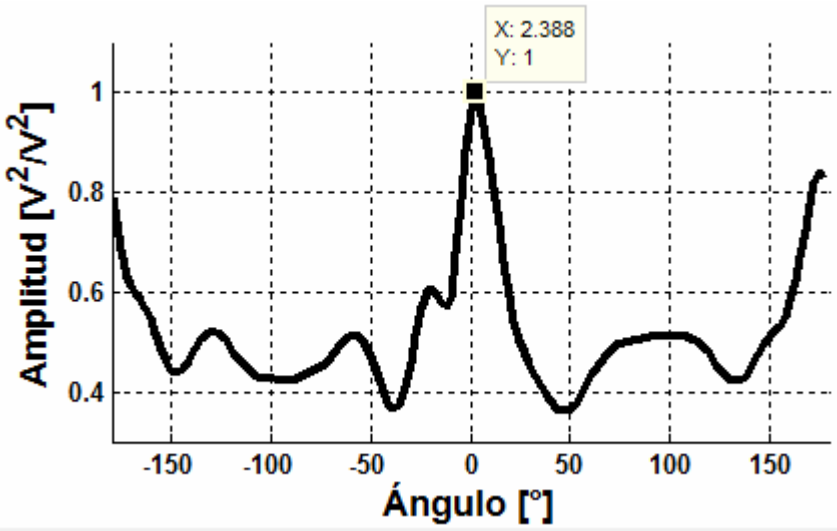


Figura 4.13. Suma de las tres señales.

La función sum() recibe como parámetros los tres apuntadores a los vectores antes mencionados. Por último con la función programada

void maximos(int *P, short tam, int *valPos);

Se obtiene el valor máximo en el vector al que apunta *P, y de tamaño igual a tam, y el valor y posición en el vector se guarda en el vector al que apunta *valPos que en el programa el código es el siguiente.

maximos(LR1, 96, valores);

Con la posición del valor máximo en el vector LR1 guardado en el vector valores, se procede a hacer la estimación del ángulo, para ello se tiene el vector ángulos de 96 muestras, el cual guarda el mapeo en ángulo de las muestras correspondientes de las funciones correlación cruzada, así que la posición donde se encuentra el valor máximo en esa posición del vector ángulos es el valor de ángulo estimado de la dirección de la fuente de voz para la ventana de 128 muestras con el método GCC sumado.

4.6.2. Método Transformada de fase PHAT

Para este método, se toman ventanas de 256 muestras pero a diferencia del programa anterior en donde los valores son valores reales, en este caso, las muestras son valores complejos, es por ello que los vectores en que se guardan las muestras correspondientes a cada canal se guardan en vectores de 512, en donde el primer valor de la muestra corresponde el primer valor del vector a su parte real y el siguiente valor a su parte imaginaria. Dado que las muestras son reales por lo que sus partes imaginarias son iguales a cero, las señales se guardan en los vectores R1, L1, y B1 de 512 localidades cada uno, correspondientes a los canales derecho, izquierdo y atrás respectivamente. En la Figura 4.14 se muestran las señales de los micrófonos.

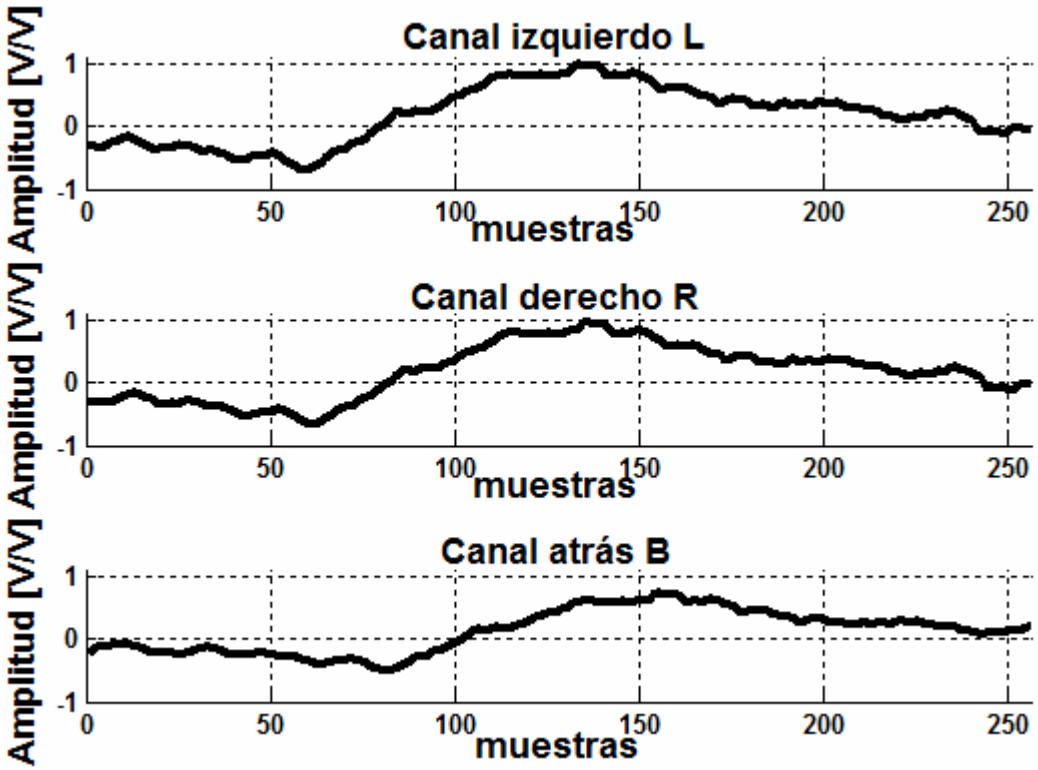


Figura 4.14. Señales en el tiempo cada una de 256 muestras.

Para el método PHAT se requiere, primero obtener el espectro de cada una de las señales, en la Figura 4.15 se muestran la parte real e imaginaria del espectro de cada canal. Para ello se calcula la transformada de Rápida de Fourier radix 2 decimada en la frecuencia, sus parámetros son el número de datos complejos del vector, la dirección del primer dato del vector y por último los valores complejos de W en formato entero Qi 15. El código de programa es el siguiente:

DSP_radix2(256, apR2, W);

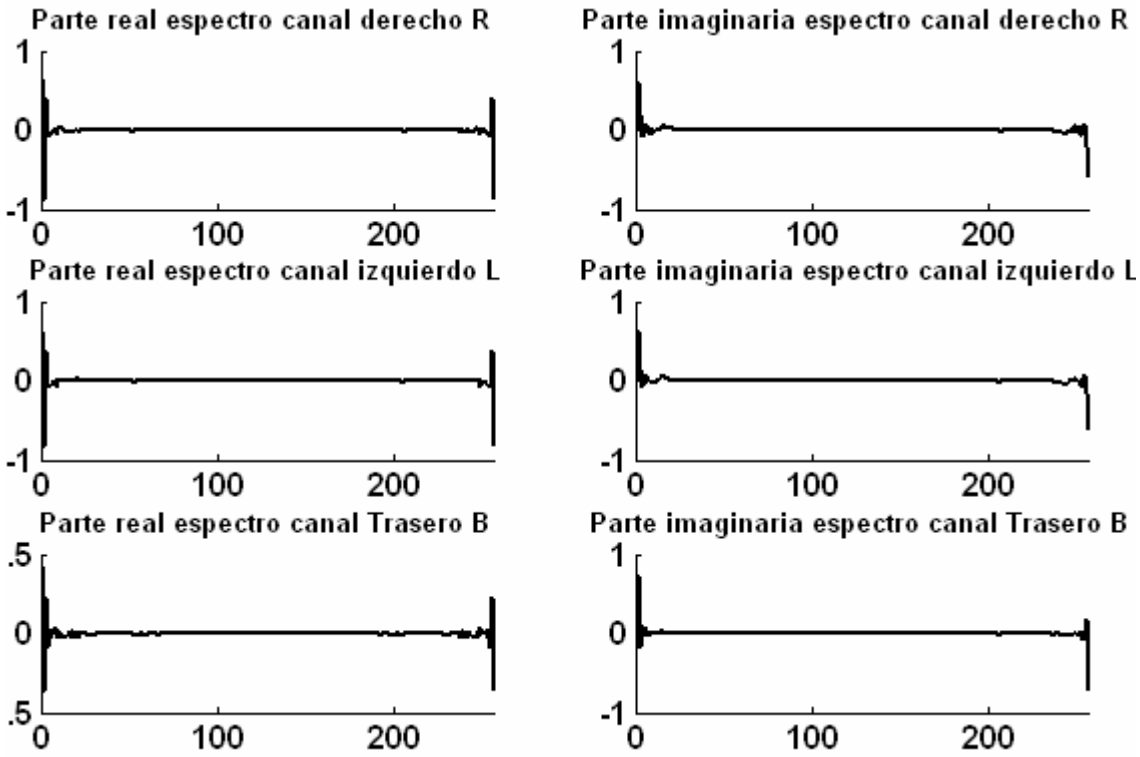


Figura 4.15. Espectro de las ventanas correspondientes de cada canal.

Son 256 datos complejos, apR2 apunta a la primera localidad del vector R, y W es el vector W complejo.

De esta manera se obtienen los tres espectros de las tres señales, dado que la función esta decimada en la frecuencia, los valores de los espectros se encuentran en desorden, para reordenar los valores se aplica la función DSP_bitrev_cplx(), la cual realiza el decimado de la función reordenando el espectro.

DSP_bitrev_cplx((int*) apR2, y, 256);

El apuntador *apR2 apunta al vector *R*, se hace un casting debido a que la función esta programada para datos tipo entero pero los datos en *R* son tipo short, y es un vector que tiene valores a reordenar, y el tercer valor es el número de valores complejos del vector a decimar. Se realiza lo mismo para las otras dos señales restantes.

Así, una vez calculados los tres espectros, se procede a realizar la transformada PHAT correspondiente a cada sub arreglo de acuerdo al capitulo 3, con la diferencia de que en lugar de realizar la anti transformada directamente de acuerdo a la ecuación (4.6) se obtiene el cuadrado, ya que para obtener PHAT de acuerdo a la ecuación es necesario obtener los módulos de los espectros lo cual implica el cálculo de raíces cuadradas lo que lleva mucho tiempo de proceso, en lugar de eso se eleva al cuadrado. En las ecuaciones (4.6) se muestra la transformada PHAT al cuadrado y (4.7) el cuadrado de la transformada PHAT para el caso de un valor complejo de *Fx*

es decir $a_x + ib_x$ y un valor complejo $a_y + ib_y$ de *Fy* donde *Fx* y *Fy* son los espectros de dos canales.

$$\left(\frac{F_X F_Y^{*}}{|F_X||F_Y|} \right)^2 \tag{4.6}$$

$$\left(\frac{(a_x + ib_x)(a_y - ib_y)}{\sqrt{a_x^{\,2} + b_x^{\,2}}\sqrt{a_y^{\,2} + b_y^{\,2}}} \right)^2$$
$$= \frac{a_x^{\,2}a_y^{\,2} + b_x^{\,2}b_y^{\,2} + 4a_x a_y b_x b_y - a_y^{\,2}b_x^{\,2} - a_x^{\,2}b_y^{\,2} + 2i(a_x a_y(a_y^{\,2} - b_y^{\,2}) + a_y b_y(b_x^{\,2} - a_x^{\,2}))}{a_x^{\,2}a_y^{\,2} + a_x^{\,2}b_y^{\,2} + b_x^{\,2}a_y^{\,2} + b_x^{\,2}b_y^{\,2}} \tag{4.7}$$

En la función transPHAT() se realiza la transformada PHAT considerando la ecuación (4.7), el código es:

transPHAT(apL2, apR2, 256, LR);

Efectúa la transformada PHAT del sub arreglo izquierdo - derecho *LR*, los apuntadores *apL2* y *apR2* apuntan a los vectores de los espectros *L* y *R*, el tercer parámetro igual a 256 indica el número de valores complejos de cada uno de los vectores, y el último parámetro indica el vector donde se guarda el resultado, en este caso es un vector con valores complejos. Se hace esto para las dos señales restantes correspondientes a sus sub arreglos. En la Figura 4.16 se muestran los espectros *LR*, *RB* y *BL* tanto su magnitud y fase, como se puede apreciar la magnitud es unitaria quedando solo la fase idealmente.

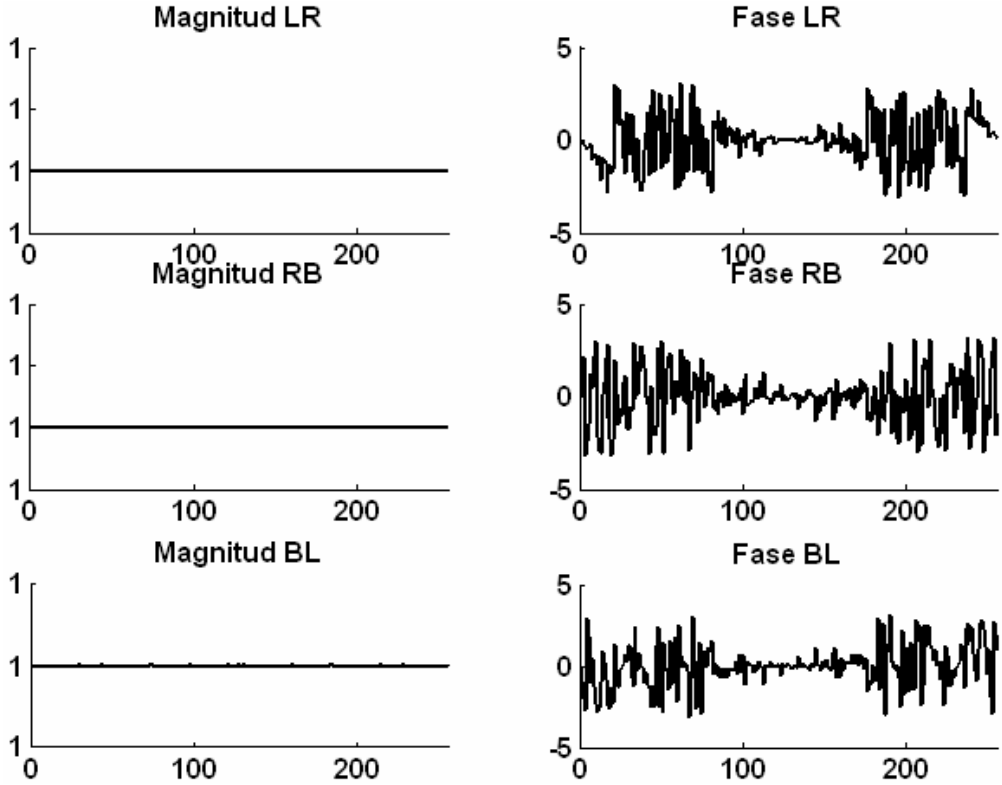


Figura 4.16. Espectros LR, RB y BL resultado de la transformada PHAT.

Se anti transforman los vectores anteriores por transformada de Fourier inversa. Para efectuar la anti transformada de Fourier a partir de la FFT se obtiene el conjugado del espectro, después se aplica la FFT y a esta nueva señal se vuelve a obtener su conjugado. Si se desea se puede normalizar la señal dividiendo por el número total de muestras, la señal resultado es la anti transformada, la función transPHAT devuelve los valores ya conjugados, por lo que a las señales nuevamente se les aplica la función DSP_radix2(), y dado que están en desorden por la decimación se reordenan aplicando la función que decima DSP_bitrev_cplx(), de estas señales no es necesario obtener su conjugado ya que se puede descartar la parte compleja y solo trabajar con la parte real. Lo anterior se realiza para cada una de las señales. En la Figura 4.17 se muestran las partes reales de las señales de correlación *lr*, *rb* y *bl* que son las anti transformadas de las señales *LR*, *RB* y *BL* resultado de la transformada PHAT.

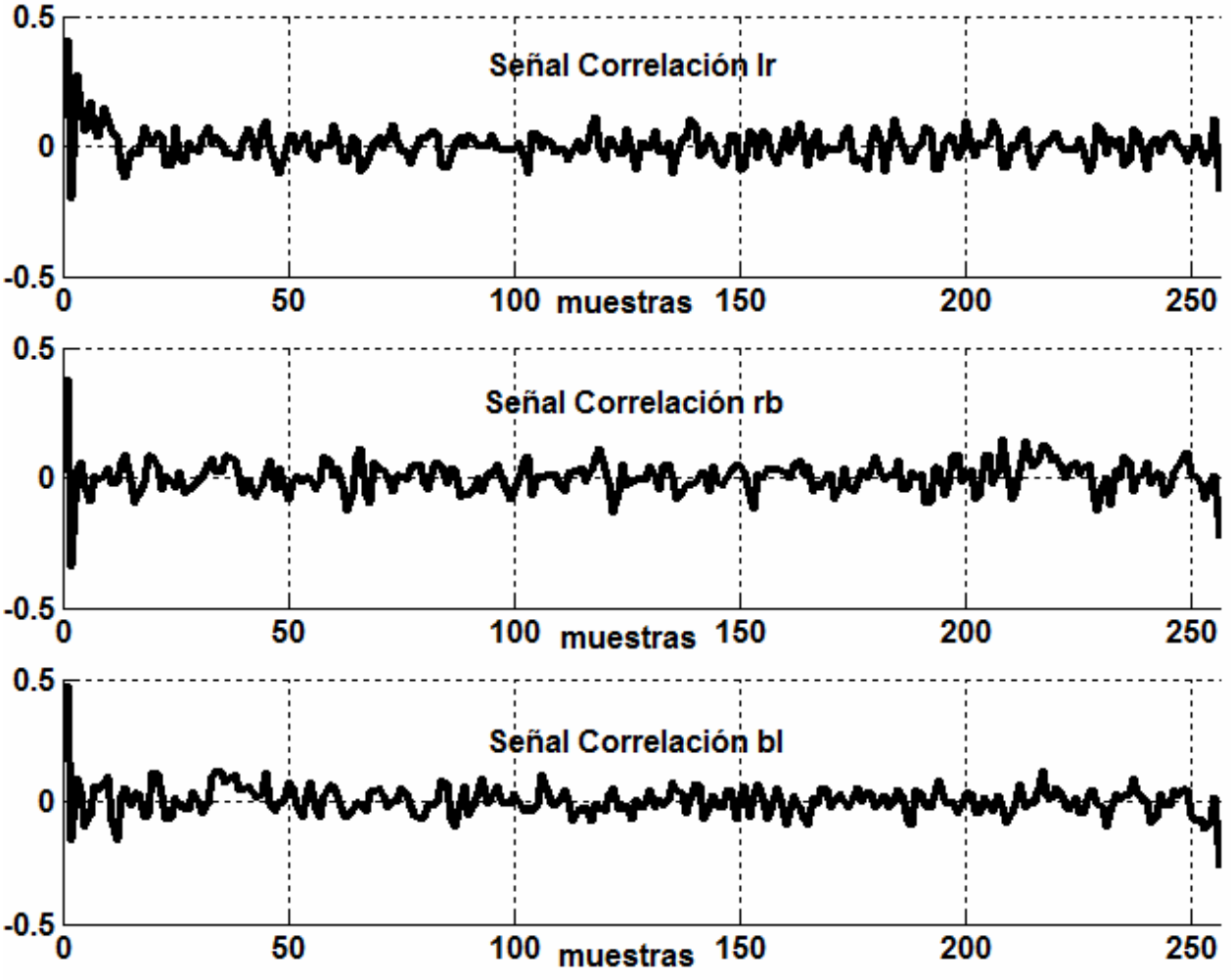


Figura 4.17. Parte real de las señales lr, rb y bl.

De las señales obtenidas se toman las ultimas 24 muestras y se concatenan con las primeras 25 muestras obteniendo un vector de 49 muestras. Las nuevas señales son ahora de 49 muestras tal como se muestra en la Figura 4.18. De ahí se ordenan y mapean de la misma forma descrita en el método GCC sumado, con las funciones reordena() y reordena2(). Ya reordenadas se suman y se guardan en un vector de lr_1 de 96 muestras. Con la función maximos() se obtiene su valor máximo y su posición y con el se obtiene el estimado de la dirección de la fuente de voz para esa ventana de 256 muestras.

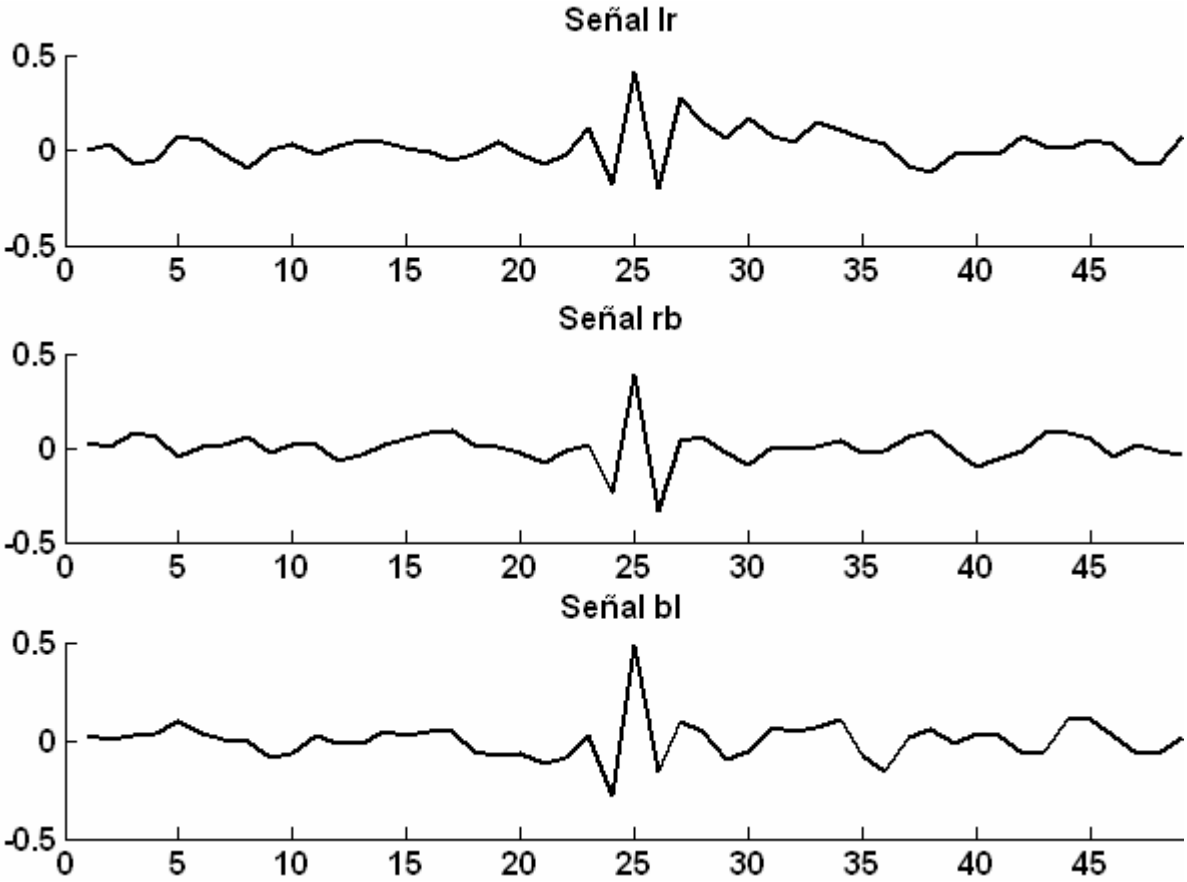


Figura 4.18. Muestras tomadas para su mapeo.

Así se obtiene un estimado de la localización de la fuente de voz cada 5.3 ms.

4.7. DIAGRAMA DE TIEMPOS

Los programas están diseñados para que puedan efectuar la estimación en tiempo real, para ello está programada la interrupción por timer 0, el timer 0 está configurado para que esté interrumpiendo cada 20.833 μs correspondiente a la frecuencia de muestreo de 48 kHz, de esta forma se puede controlar la frecuencia de muestreo a la que se estarían tomando las muestras de un convertidor externo a la tarjeta de desarrollo DSK TMS320C6416, cada vez que se presenta una interrupción por timer se puede solicitar una conversión de los tres canales, esperar hasta que estén listos los datos y guardar en memoria.

En la Figura 4.19.a) se muestra como se guardan los buffer de datos, se tienen los dos vectores para cada canal *L1*, *R1*, *B1*, *L2*, *R2* y *B2*, al iniciar el programa se van llenando primero los vectores *L1*, *R1* y *B1*, el llenado se hace en cada interrupción. Hasta este momento ningún vector es procesado, una vez que los primeros vectores se llenan se tiene la primera ventana para procesar, así que se procede a llenar los vectores *L2*, *R2* y *B2*.

Ya que se tiene la primera ventana en el programa principal, ésta se procesa como se puede apreciar en la Figura 4.19.b) mientras en cada interrupción se van llenando los vectores *L2*,

R2 y *B2*. El proceso de la primera ventana termina antes de que se complete la ventana 2, de esta forma no se corrompen los datos y se obtiene el primer estimado de la dirección de la fuente de sonido el cual se guarda para su análisis.

El programa principal espera hasta que se complete la ventana 2 como se observa en la Figura 4.21.c) una vez terminada, inicia el procesamiento de la ventana 2 y ahora se empieza a capturar la ventana 3 en *L1, R1* y *B1*. Así continua el proceso de forma cíclica.

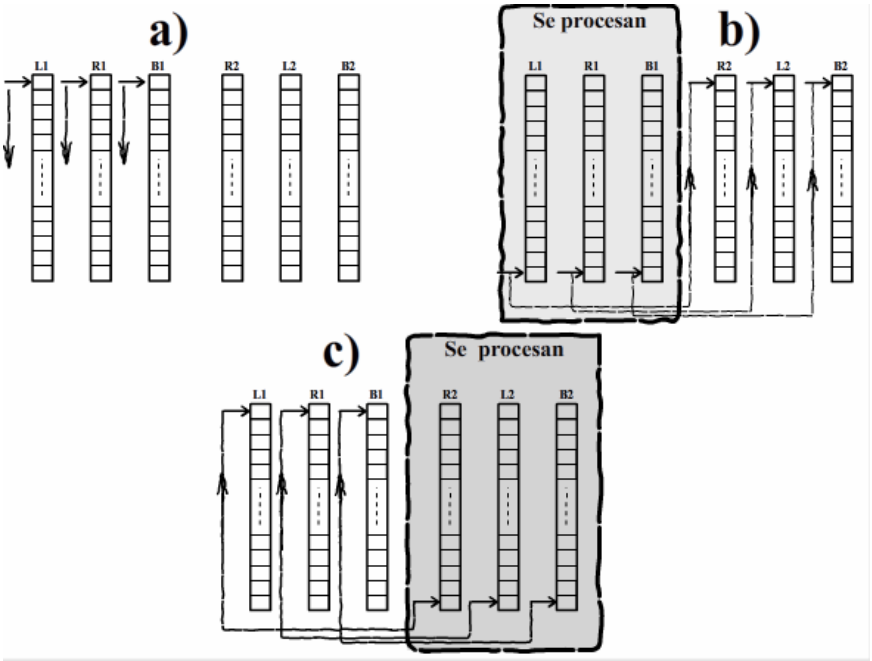


Figura 4.19. Captura de datos en vectores, orden de captura y procesamiento.

El proceso es automático ya que se considera que una fuente de voz se encuentra presente en todo momento. Para que el proceso se inicie y detenga conforme se presente una fuente de voz se necesita de un detector de inicio y fin de palabra.

Dependiendo del método se elige el tamaño de ventana originalmente, se pensó que ambos métodos ocuparan un tamaño de ventana de 128 muestras. En el caso de GCC Sumado es suficiente ya que el proceso de la ventana se realiza antes de que la siguiente ventana esté totalmente capturada, pero en el caso de transformada PHAT si se elegía una ventana de 128 muestras el proceso sobrepasa el tiempo que tarda en obtenerse la siguiente ventana, por esta razón se eligió una ventana de 256 muestras ya que aunque son mas muestras, el procesamiento se hace antes de que se tenga la siguiente ventana.

En la Figura 4.22 Se muestra un diagrama de tiempos, los pequeños puntos representan las interrupciones que definen el período de muestreo, en donde τ_w es el tiempo necesario para capturar una ventana. Dado que las señales se muestrean a 48 kHz para 128 muestras son aproximadamente 2.7 ms, y para 256 muestras son aproximadamente 5.3 ms, así que el procesamiento de las ventanas τ_p se deben realizar en un tiempo menor a τ_w. Al terminar el

procesamiento de ventana se tiene un estimado de la dirección de la fuente de voz para la ventana 1 es E_1 y para la ventana 2 es E_2 estos valores estimados se guardan para su análisis.

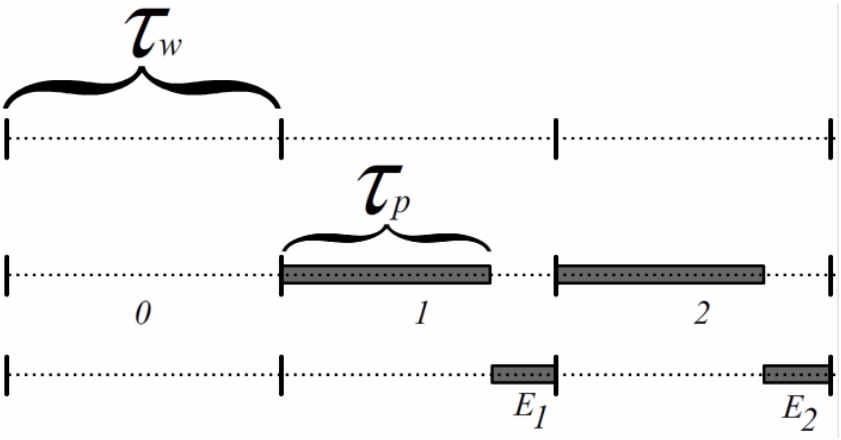


Figura 4.22. Tiempo de ventana, tiempo de proceso.

4.8. SÍNTESIS

Al principio del capítulo se describe de forma general como está constituido el sistema, de acuerdo al orden en que se listan las partes del sistema se van describiendo cada tema. Se describe brevemente la arquitectura del DSP TMS320C6416 que es la arquitectura para el procesamiento de datos elegida. Continuando con los criterios tomados en cuenta para la construcción del arreglo desde la elección de los micrófonos em-926, su polarización y el diseño del circuito impreso en cual están colocados.

Después, se explica el diseño del circuito acondicionador de señal, construido a partir de amplificadores operacional, cuya función es amplificar, limitar en banda y dar un voltaje de offset necesario para la conversión analógico digital. Después se describe la conversión que se realiza con la tarjeta de desarrollo DSK TMS320F2812 de la cual se extraen los datos en forma de un archivo.

Con los archivos obtenidos se insertan los datos en la memoria del DSP TMS320C6416 a través del entorno de desarrollo Code Componer Studio.

Ya por último, se describen los dos algoritmos programados para su caracterización y que ejecuta el DSP TMS320C6416, el algoritmo GCC sumado y transformada PHAT. Se detallan los criterios tomados para la elección de tamaño de ventana de datos a procesar, los tiempos de procesamiento así como el control del programa y las funciones programadas para el procesamiento de las señales.

5

PRUEBAS Y RESULTADOS

En este capítulo se presentan las pruebas resultado del sistema, la descripción de la operación del equipo utilizado, la colocación del equipo, y la manipulación de los datos obtenidos para la presentación de resultados.

5.1. UBICACIÓN DEL EQUIPO

El arreglo de micrófonos se coloca sobre un tripié, el cual tiene adaptado un disco que indica los grados, el centro del disco transportador coincide con el centro del arreglo. El disco transportador se mantiene fijo, mientras el arreglo de micrófonos puede girar sobre el disco transportador.

Para simular una fuente de voz se utiliza una bocina la cual emite una grabación, para cambiar la dirección de la fuente, se hace rotar el arreglo de micrófonos colocando al ángulo al cual se quiere que las ondas de la fuente de voz incidan sobre el arreglo.

La bocina se coloca a tres metros de distancia del arreglo de micrófonos y con ayuda de un apuntador laser que se alinea con el centro del transportador, de esta forma se mide el ángulo formado por el eje central de la bocina y la recta perpendicular al arreglo de micrófonos.

5.2. OBTENCIÓN DE MUESTRAS

El arreglo está conectado al circuito acondicionador de señal, que a su vez se conecta con los canales del ADC de la tarjeta de desarrollo DSK TMS320F2812, dentro del programa y con ayuda del ambiente Code Composer Studio (CCS), se coloca un breakpoint en el punto donde se terminan de capturar Mientras la bocina está emitiendo la señal que incide sobre el arreglo, se capturan las muestras haciendo correr el programa, al llegar al breakpoint se guardan los datos de memoria dato del DSP,

los vectores correspondientes a cada canal, cabe mencionar que el programa captura 2731 muestras por canal. TMS320F2812 de la localidad 0x8000 a 0x9FFE con ayuda de CCS, se almacenan en un archivo con terminación .dat en formato hexadecimal. Los datos son valores enteros signados con Qi igual a 11.

Para efecto de la presentación de resultados se requiere tomar los datos para un ángulo de incidencia distinto, se ajusta el arreglo para ese ángulo y se ejecuta el procedimiento anterior. Ya generado el archivo se ordenan los datos mediante, es decir, la cual ordena estos vectores de 2731 muestras en otros 21 archivos los cuales corresponden a 21 ventanas esto es para el caso del algoritmo de GCC sumado.

Para el caso de transformada PHAT se ordena los datos en vectores complejos de 256 muestras, parte real contigua a su parte imaginaria, por lo que se generan 11 archivos correspondientes a las 11 ventanas que se pueden obtener de 2731 muestras.

5.3. CORRELACIÓN CRUZADA DE VENTANAS

Los datos son guardados en memoria del DSP TMS320C6416, de una ventana generada en el momento en que incide sobre el arreglo una señal de audio a una dirección de -90°. Los datos se guardan a partir de una localidad de memoria donde corresponde la localidad del vector R1 contiguo a L1 y B1 correspondientes a los tres canales.

Los datos se almacenan en variables enteras tipo short de 16 bits. En la Figura 5.1 se aprecian sus correspondientes gráficas, las cuales se generan en CCS.

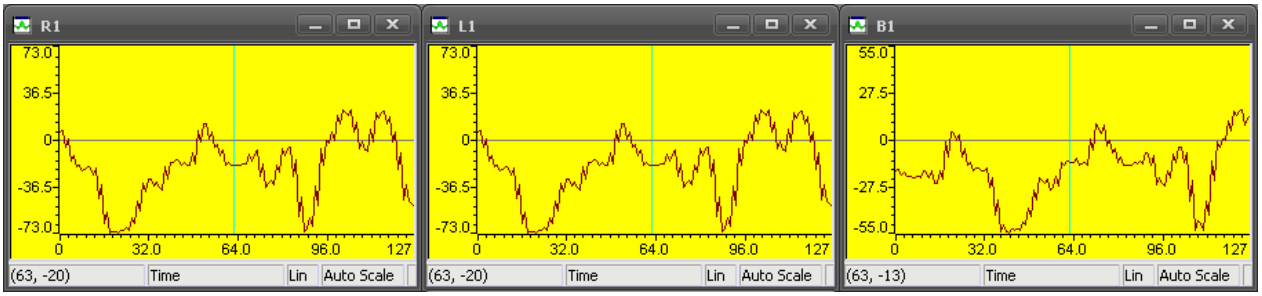


Figura 5.1. Gráficas en el tiempo de ventanas de 128 muestras de los canales R, L, y B.

Con estas señales se realiza la correlación cruzada entre canales para obtener las señales LR, RB y BL correspondientes a los tres sub arreglos. Dado que la información más relevante de la correlación cruzada en este caso se encuentra en las 49 muestras centrales solo se realiza la operación correlación cruzada para obtener estas muestras y así ahorrar memoria y tiempo de procesamiento. En la Figura 5.2 se muestran las gráficas de correlación LR, RB y BL.

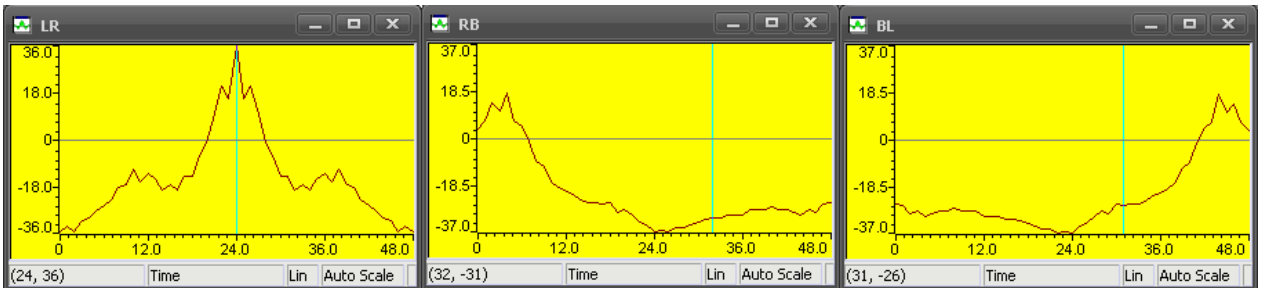


Figura 5.2. Gráficas de las 49 muestras centrales de la correlación cruzada entre canal, señales LR, RB y BL.

Las señales de correlación cruzada LR, RB y BL son datos de tipo entero, es decir de 32 bits a diferencia de las señales de cada canal que son de 16 bits. Dado que las señales se pueden mapear de -90° a 90° y considerando que las señales son simétricas se generan las señales LR1, RB1 y BL1, las cuales corresponden al intervalo de -180° a 180° aunque relativos a su correspondiente sub arreglo, y constan de 96 muestras. En la Figura 5.3 se muestra estas nuevas señales. Estas señales son resultado de la función reordena() programada por usuario.

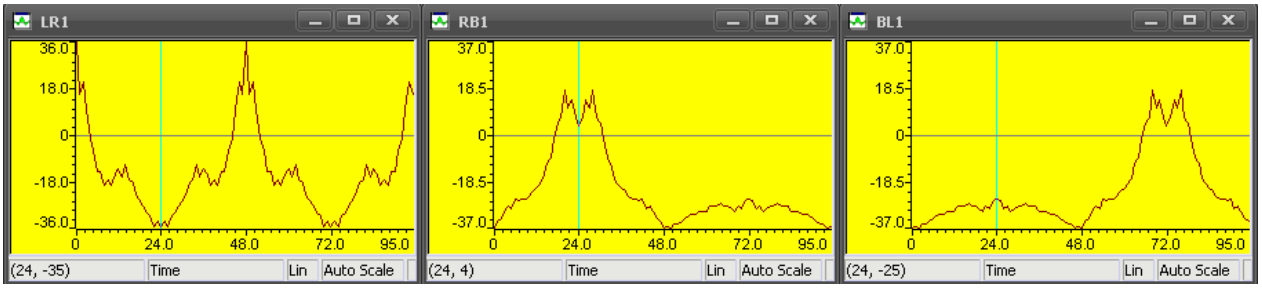


Figura 5.3. Señales LR1, RB1 y BL1.

Dado que las señales están referenciadas a un ángulo relativo a su correspondiente sub arreglo es necesario realizar un corrimiento de las señales RB1 y BL1, en cambio la señal LR1 no es necesario realizar ningún corrimiento ya que todo se referencia con respecto a su sub arreglo. Como se mencionó en el capitulo anterior la función suma considerando los corrimientos de las dos señales antes mencionadas y se van guardando en las localidades correspondientes a la señal LR1. El resultado en la Figura 5.4 más adelante se somete a una función de mapeo, la fuente se encuentra a cero grados por lo que el valor máximo está en la muestra 48.

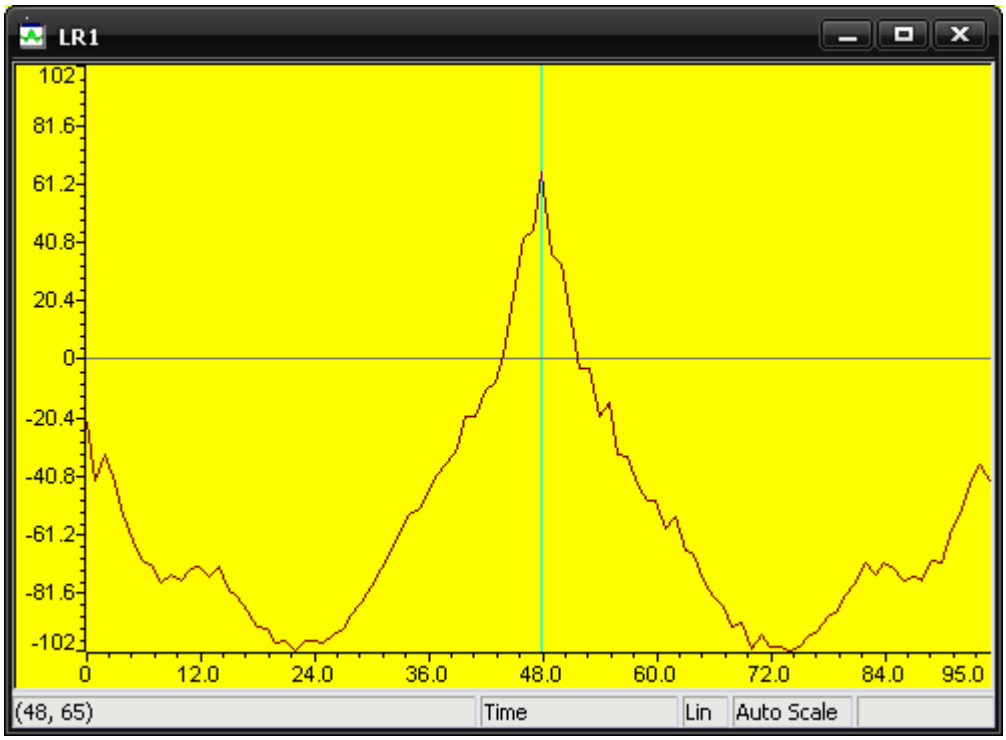


Figura 5.4. Señal resultado del método GCC Sumado.

La señal resultado es la GCC sumado, que dependiendo de la ubicación de su valor máximo será la posición angular de la fuente de audio a detectar.

5.4. MAPEO CON EL MÉTODO GCC SUMADO

A cada una de las 96 muestras de la señal GCC sumado les corresponde un valor de posición angular, en la Figura 5.5 se muestra la función mapeo la cual consta de 96 muestras en punto flotante correspondientes a los valores en grados de posición angular de la fuente, desde -180° a 180°.

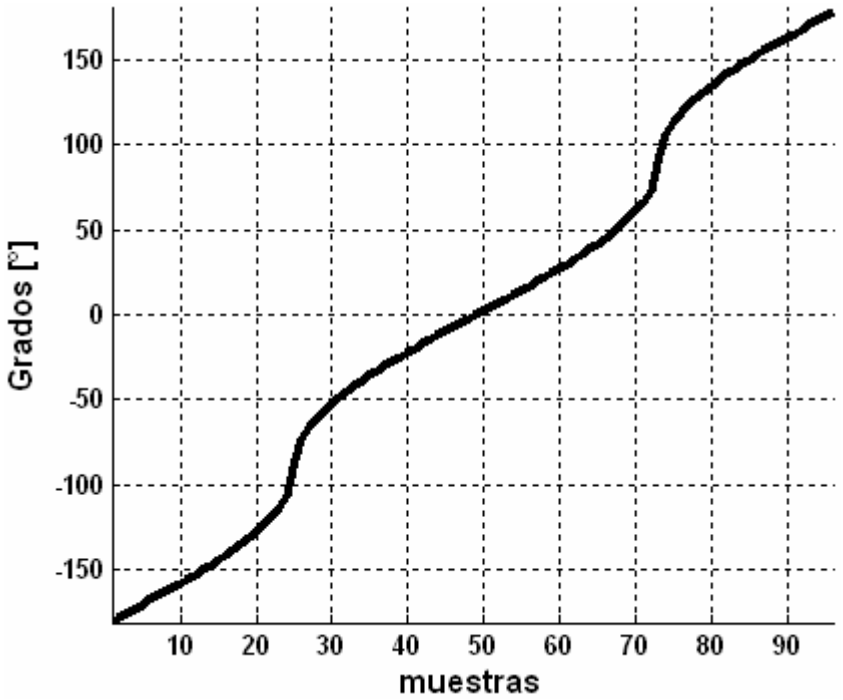


Figura 5.5. Función de Mapeo, muestra posición angular en grados.

Con la función maximos() se busca el valor máximo de la señal GCC sumado, y se busca en la función mapeo el valor en ángulo correspondiente de esta forma obtener un estimado, tal como se muestra en la Figura 5.6. Se puede observar que en la parte inferior se encuentra la función GCC sumado resultado del algoritmo que consiste de 96 muestra a la cual corresponde la función mapeo que también consiste de 96 muestras.

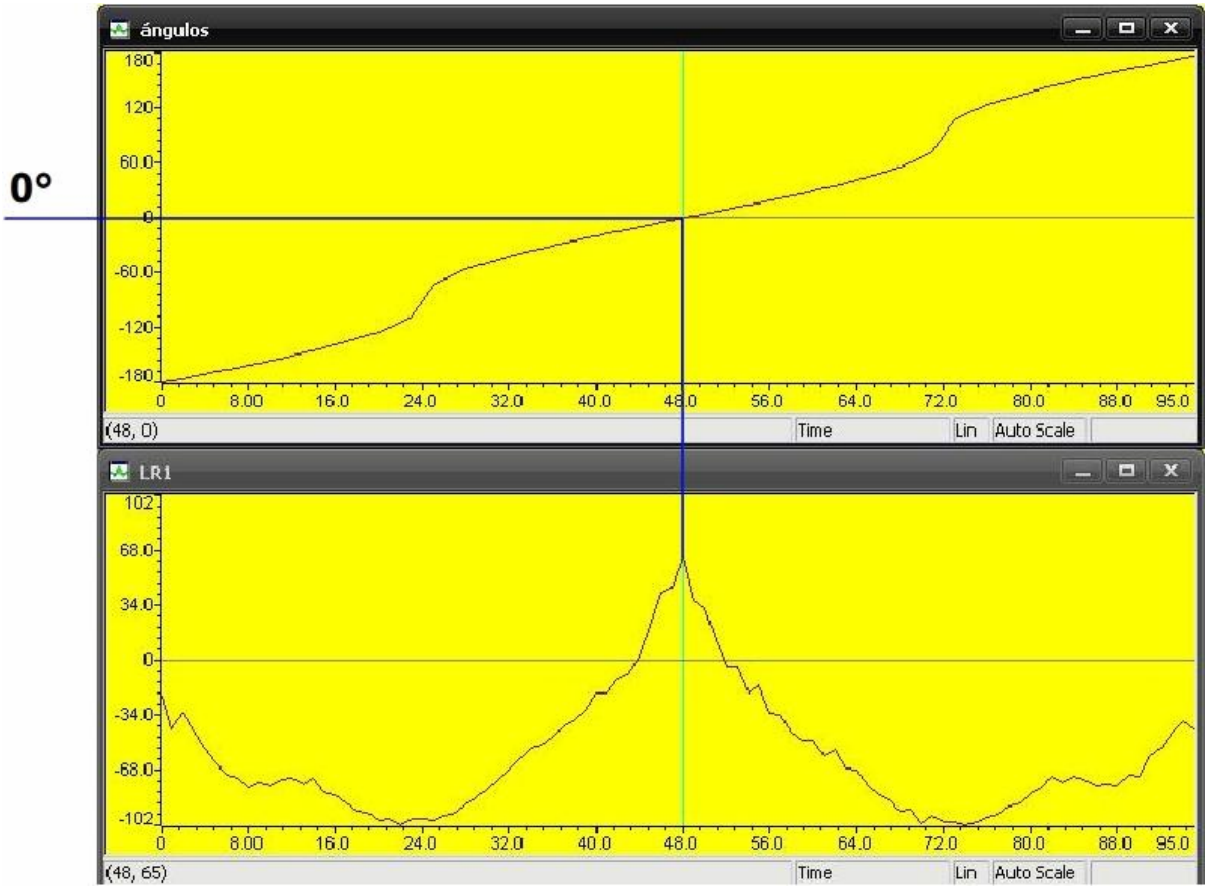


Figura 5.6. Función GCC sumado y función mapeo, estimado de dirección de fuente de voz.

Como se puede observar, el valor máximo se encuentra en la muestra 48 lo cual en la función mapeo tiene un valor de 0, lo cual corresponde a los 0° en los que se encuentra la fuente de sonido. El valor se guarda en la variable tipo flotante ESTIMADO como se puede ver en la Figura 5.7.

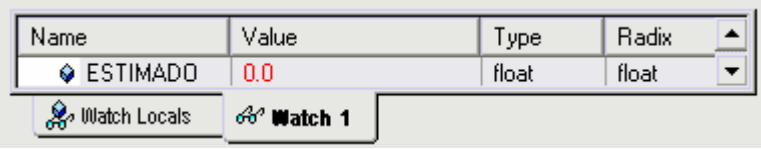


Figura 5.7. Valor estimado de la dirección angular de la fuente de voz.

5.5. APLICACIÓN DE LA TRANSFORMADA PHAT A VENTANAS

En la Figura 5.8 se muestran las gráficas de los tres canales de 256 muestras.

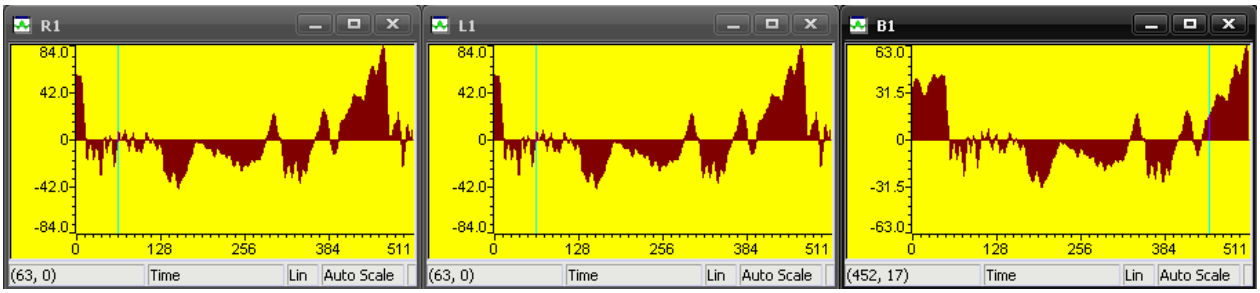


Figura 5.8. Señales correspondientes a los tres canales L, R, y B.

A diferencia del método GCC sumado donde se trabaja exclusivamente con la parte real, en este caso los datos son complejos. Sus espectros se muestran en la Figura 5.9, y fueron calculados por la transformada rápida de Fourier radix 2.

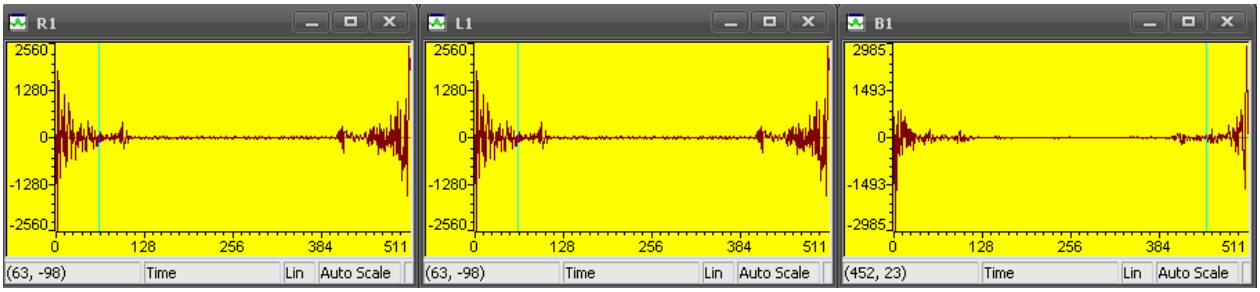


Figura 5.9. Espectros de las señales L, R y B, nombrados L1, R1 y B1.

Se les aplica la transformada PHAT a las señales L, R y B, de lo cual se obtienen tres señales correspondientes a cada sub arreglo LR, RB, y BL. Como se muestra en la Figura 5.10.

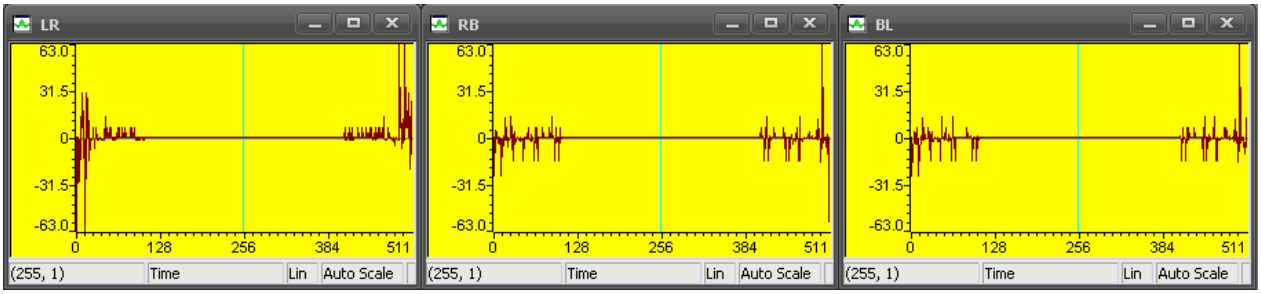


Figura 5.10. Señales Transformadas de fase PHAT LR, RB y BL.

De estas señales se obtiene su representación en el tiempo aplicando la antitransformada de Fourier como se muestra en la Figura 5.11.

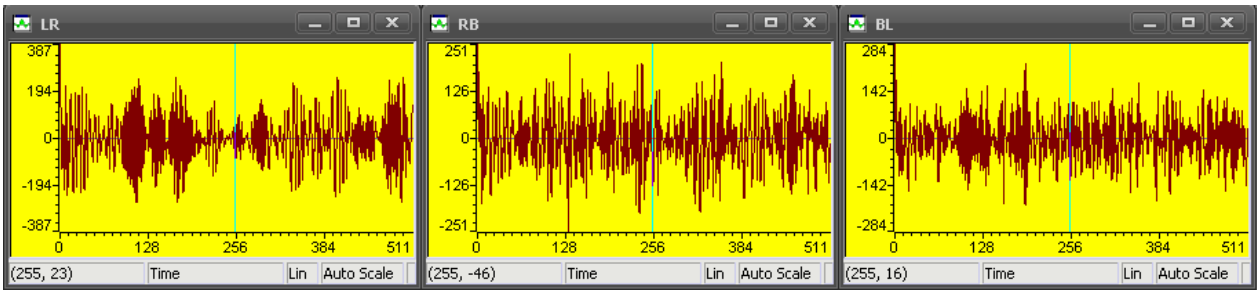


Figura 5.11. Señales después de aplicarles la antitransformada.

De estas señales se reordenan y se obtienen las últimas 24 muestras de la parte real y las primeras 25 parte real para obtener información del valor máximo donde se encuentra la fuente como se muestra en la Figura 5.12. Las señales lr, rb, y bl.

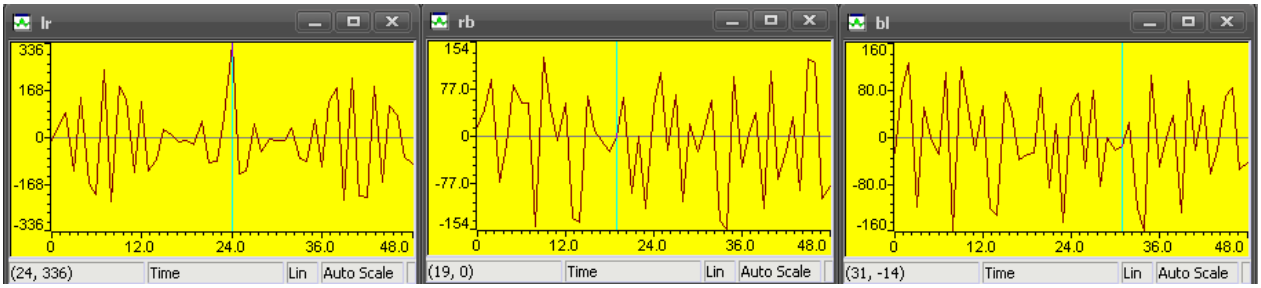


Figura 5.12. Señales con 49 muestras correspondientes a aquellas muestras con información del retraso entre señales.

Las señales de la Figura 5.12 se reordenan para obtener señales de 96 muestras y como se muestra en la Figura 5.13. las señales lr1, rb1 y bl1.

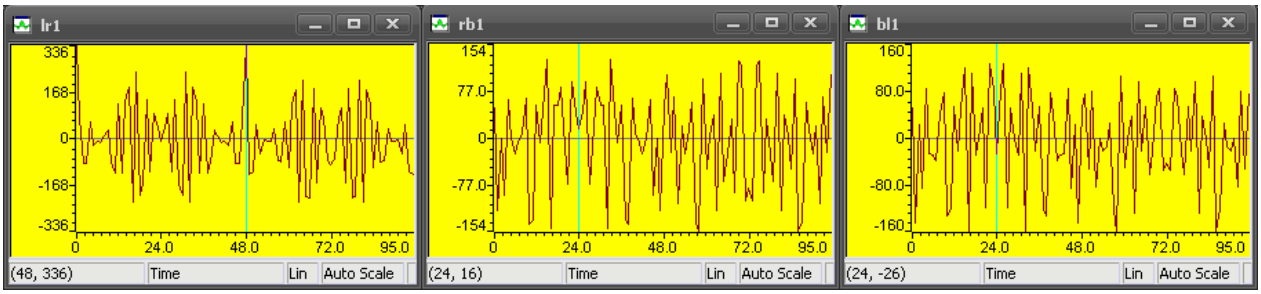


Figura 5.13. Señales de 96 muestras lr1, rb1 y bl1.

5.6. MAPEO CON EL MÉTODO TRANSFORMADA DE FASE PHAT

De la Figura 5.13 se suman las muestras de forma ordenada como se explicó en el capitulo 4 y se mapean como en la Figura 5.14. En la parte inferior se muestra el resultado de la transformada PHAT en la ventana lr1, del cual se obtiene el valor máximo y se obtiene el estimado del ángulo de la ventana ángulos ubicada en la parte superior.

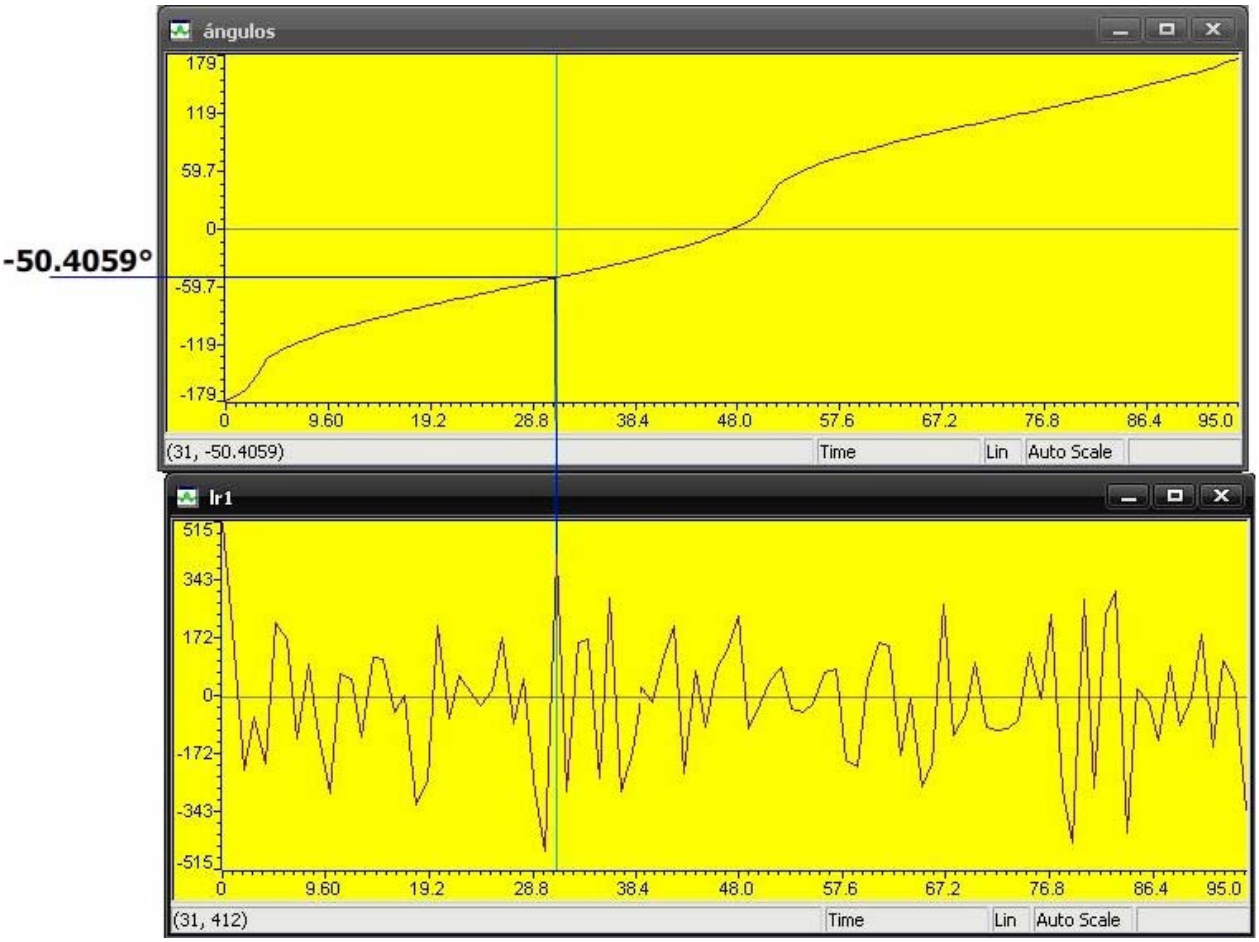


Figura 5.14. Mapeo de la función resultado de transformada PHAT.

Como se puede observar el valor máximo se encuentra en la muestra 31 a la cual le corresponde el valor de -50.4059° de acuerdo con la función mapeo en la parte superior.

5.7. GRÁFICAS RESULTADO

En las siguientes gráficas se muestran los resultados con método GCC y método transformada PHAT. Cabe mencionar que todas las graficas se encuentran normalizadas con respecto al valor máximo de los tres canales o las tres correlaciones cruzadas.

5.7.1. Resultados con Método GCC

En las Figuras 5.15, 5.16, 5.17 y 5.18 muestran las gráficas resultado con Método GCC obtenidas con datos reales con GCC, en las que se observa el proceso del método GCC. En este caso sobre el arreglo incide una señal de voz a 0° con lo que el valor estimado es de 0°

En la Figura 5.15 se muestra las señales capturadas por cada micrófono de 128 muestras a una frecuencia de muestreo de 48 kHz, identificadas como R1, L1 y B1.

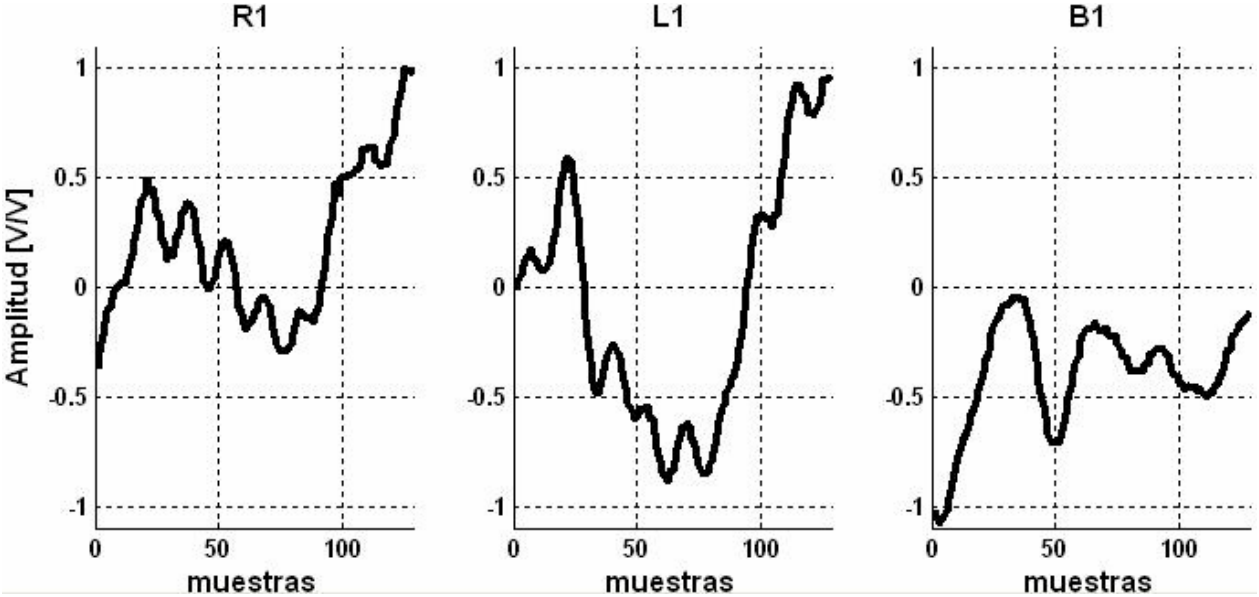


Figura 5.15. Señales de 128 muestras de los canales derecho, izquierdo y atrás.

En la Figura 5.16 se muestran las 49 muestras centrales de las correlaciones cruzadas entre las tres señales, correspondientes a los tres sub arreglos identificados como LR, RB y BL, estas muestras son las que ofrecen mayor información con respecto a la localización angular de la fuente de sonido.

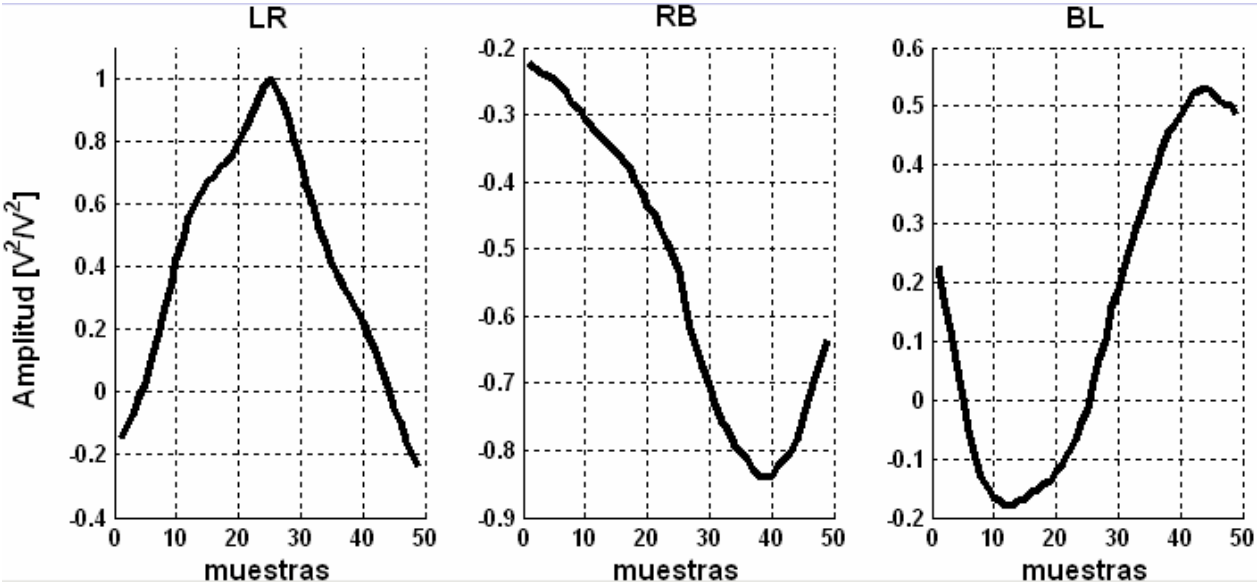


Figura 5.16. Señales correlación cruzada correspondientes a los sub arreglos LR, RB y BL.

Debido al fenómeno llamado cono de confusión se forma la señal de 96 muestras considerando que hay intervalos simétricos tal como se muestra en la Figura 5.17 y de acuerdo a lo que se explica en el capítulo 4.

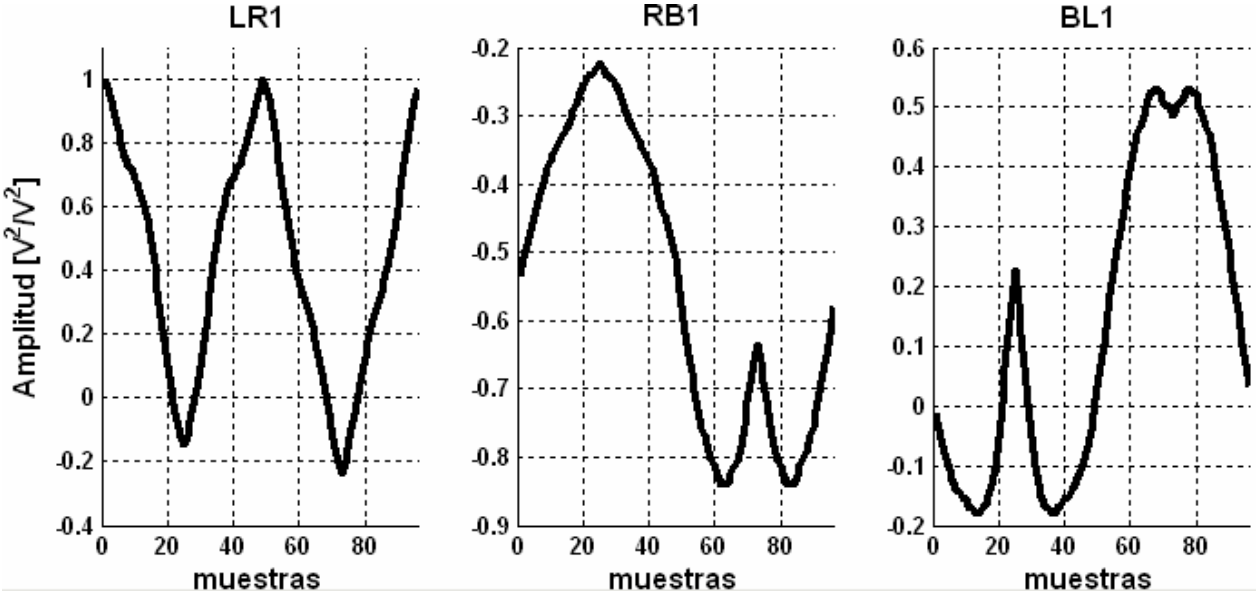


Figura 5.17. Señales reordenadas para que correspondan al intervalo angular de -180° a 180°.

En la Figura 5.18 se muestra la suma de las tres señales correlación cruzada de la Figura 5.17 considerando los corrimientos de estas por el corrimiento angular, como se explica en el capítulo 4.

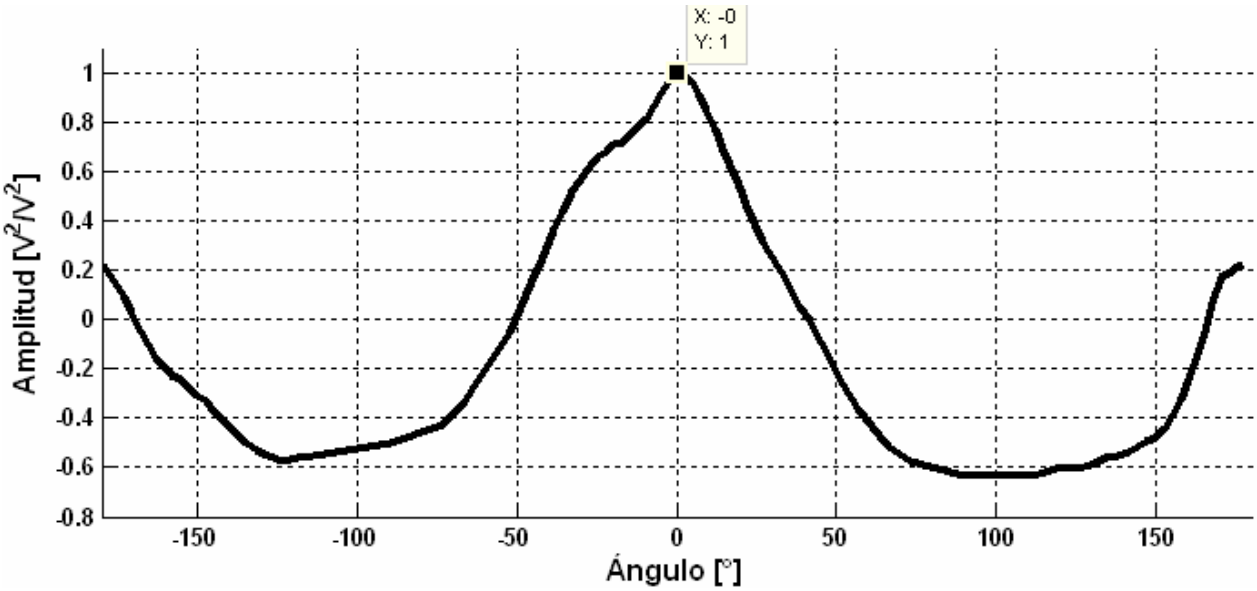


Figura 5.18. Suma de las señales y obtención del valor estimado del ángulo correspondiente al valor máximo.

En la Figura 5.18 se observa que el estimado es en 0°, por lo que en este caso se tiene un error de casi 0%.

5.7.2. Resultados con Método transformada PHAT.

En las Figuras 5.19, 5.20, 5.21 y 5.22 se muestran el proceso del método transformada PHAT de acuerdo a la explicación dada en el capítulo 4.

En la Figura 5.19 se observa las señales de entrada en el tiempo de 256 muestras de los canales derecho R1, izquierdo L1 y atrás B1.

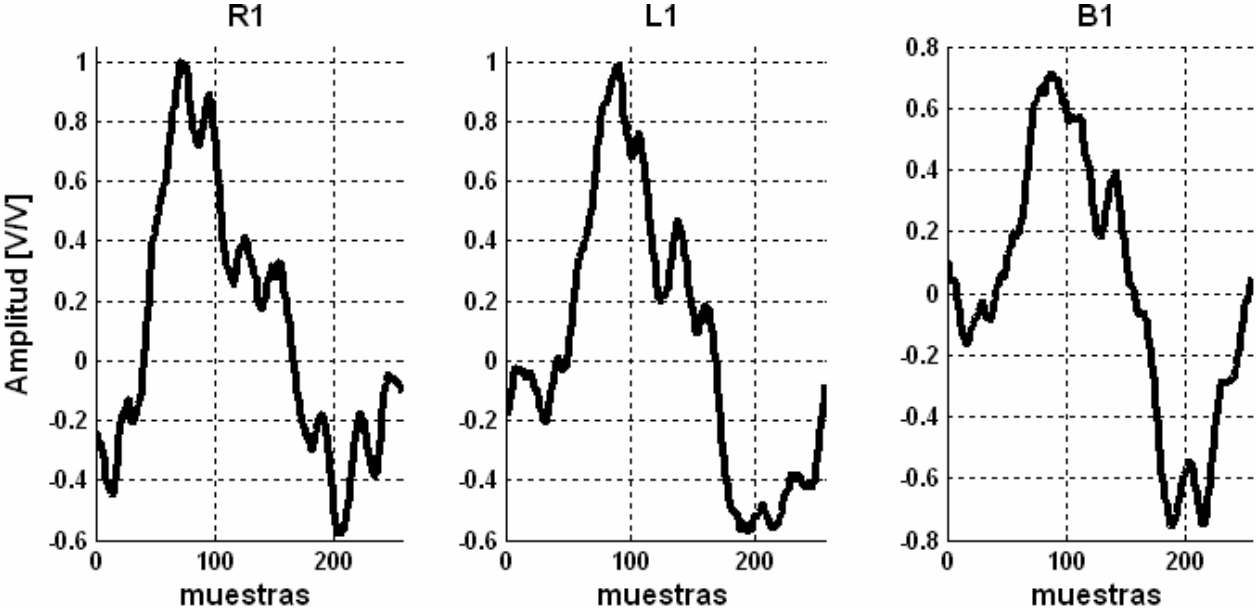


Figura 5.19. Señales R1, L1 y B1 de 512 muestras.

En la Figura 5.20 se observan las señales que contienen información de la localización del ángulo de 49 muestras resultado de la transformada PHAT.

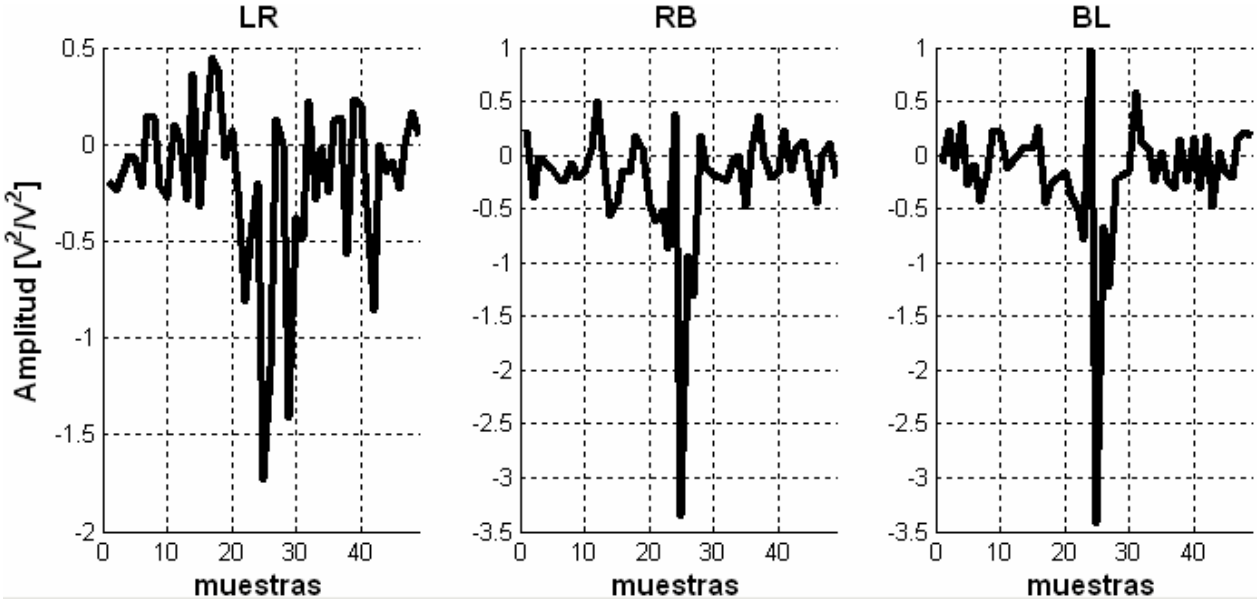


Figura 5.20. Parte de las señales resultado de los resultados de la transformada PHAT.

En la Figura 5.21 se muestran las señales mapeadas de 180° a -180° relativos a cada sub arreglo.

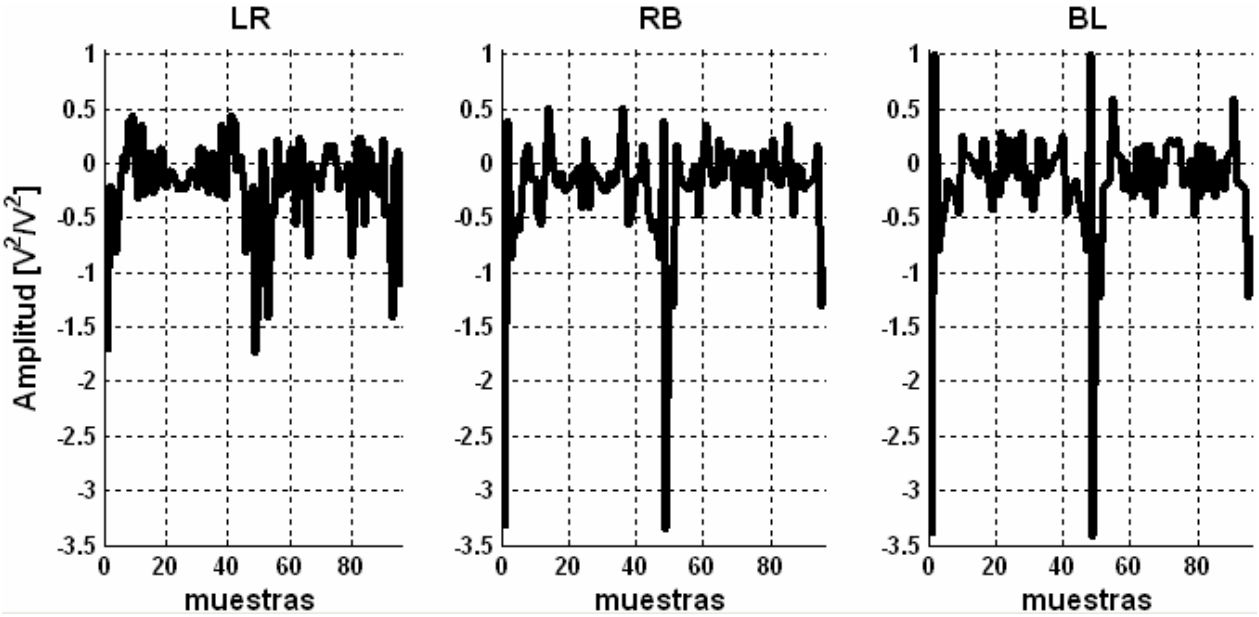


Figura 5.21. Señales re ordenadas para que correspondan al intervalo angular de -180° a 180°.

Y por último en la Figura 5.22 se muestra la suma de las tres señales mapeadas de 180° a -180° y el estimado, en este caso la fuente de voz se encuentra a 40° del centro del arreglo y el valor estimado es de 35.69°, un error porcentual del 10.78 %.

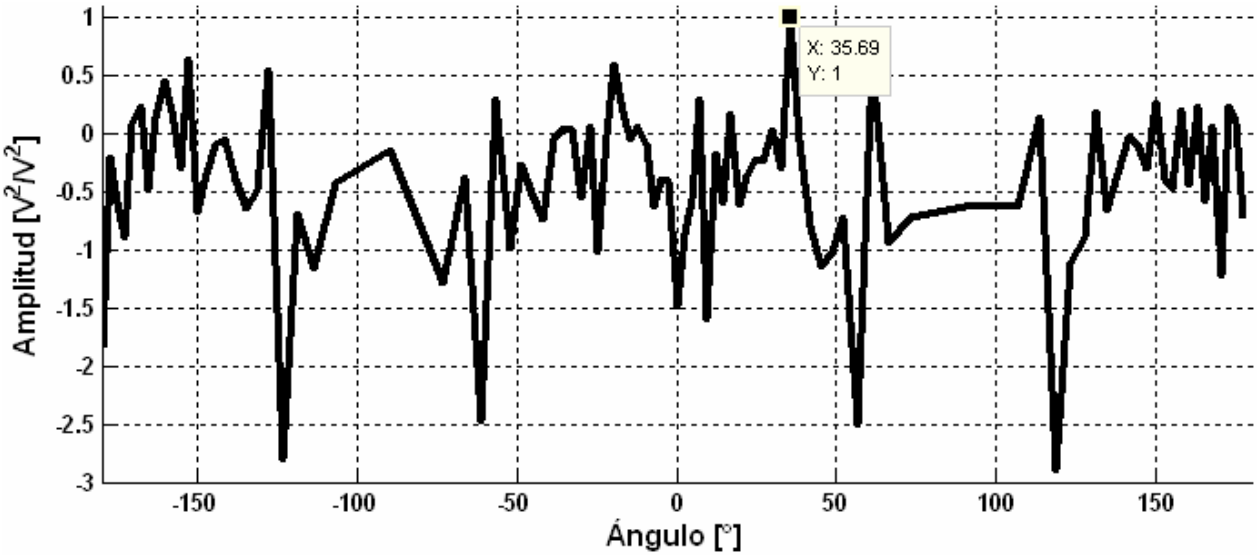


Figura 5.22. Suma de las señales y obtención del valor estimado del ángulo correspondiente al valor máximo.

5.8. ESTIMACIONES

En la tabla 5.1 se muestran los valores estimados de las simulaciones con GCC para variaciones de 5° de -180° a 180°. Las señales se ingresan al DSP por medio de archivos aunque en este caso las señales son sintéticas, el defasamiento entre estas en lugar que se realice en el espacio se defasan por filtrado, pero el procesamiento se efectúa en el DSP.

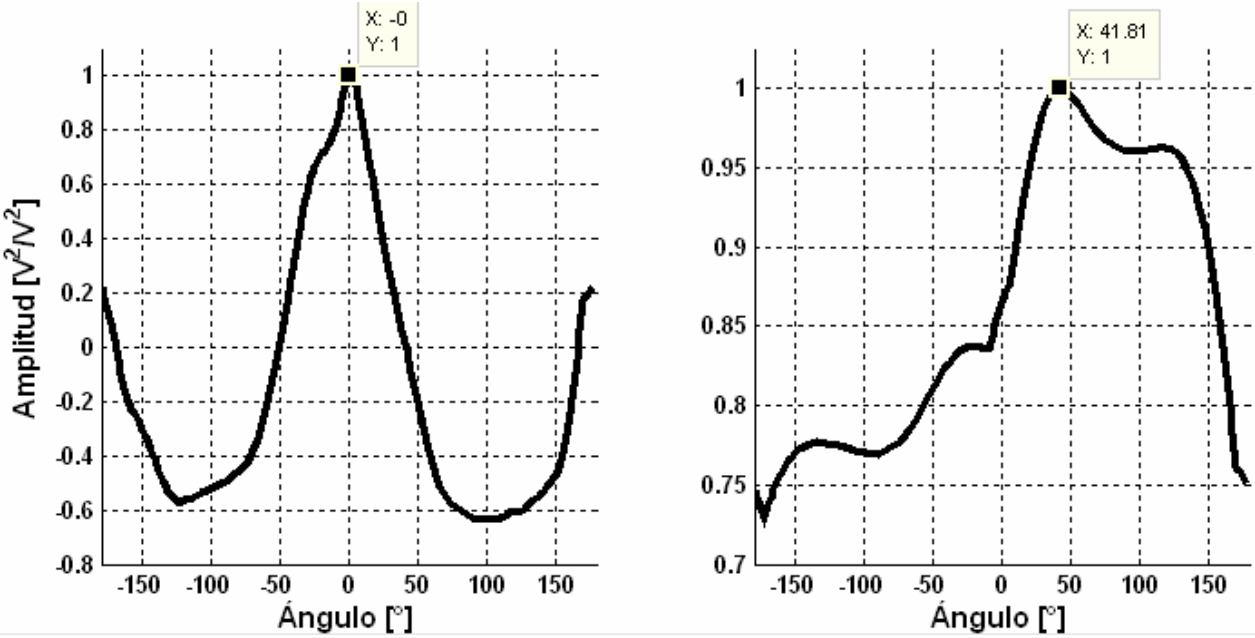
Ángulo	Estimado GCC	Ángulo	Estimado GCC
-180°	-180°	0°	-4.7802°
-175°	-180°	5°	0°
-170°	-175.2198°	10°	4.7802°
-165°	-170.4059°	15°	9.5941°
-160°	-165.5225°	20°	14.4775°
-155°	-165.5225°	25°	19.4712°
-150°	-160.5288°	30°	24.6243°
-145°	-155.3757°	35°	30°
-140°	-150°	40°	35.6853°
-135°	-144.3147°	45°	41.8103°
-130°	-138.4096°	50°	41.8103°
-125°	-131.4096°	55°	48.5904°
-120°	-123.5573°	60°	56.4427°
-115°	-113.5565°	65°	66.4435°
-110°	-90°	70°	66.4435°
-105°	-66.4435°	75°	113.5565°
-100°	-66.4435°	80°	113.5565°
-95°	-56.4427°	85°	113.5565°
-90°	-41.8103°	90°	118.955°
-85°	-41.8103°	95°	118.955°
-80°	-113.5565°	100°	45.0995°
-75°	-113.5565°	105°	52.3415°
-70°	-113.5565°	110°	61.045°
-65°	-66.4435°	115°	73.4022°
-60°	-66.4435°	120°	106.5978°
-55°	-56.4427°	125°	118.955°
-50°	-56.4427°	130°	127.6585°
-45°	-48.5904°	135°	134.9005°
-40°	-41.8103°	140°	141.3178°
-35°	-35.6853°	145°	147.2028°
-30°	-27.2796°	150°	157.9755°
-25°	-22.0245°	155°	163.0422°
-20°	-27.2796°	160°	165.5225°
-15°	-14.4775°	165°	170.4059°
-10°	-14.4775°	170°	170.4059°
-5°	-9.5941°	175°	175.2198°

Tabla 5.1. Valores estimados con los métodos GCC y transformada PHAT con señales sintéticas.

De la Tabla 5.1, se observa que la detección con señales sintéticas que se efectúa para el método GCC corriendo en la tarjeta de desarrollo DSK TMS320C6416, no se puede efectuar la detección correcta en los intervalos de -70° a -105° y de 75° a 115° y en cambio en los intervalos de -180° a -115°, -65° a 70° y de 120° a 175° se puede obtener una estimación bastante

aproximada al valor real. También se hizo la misma prueba con el método PHAT aunque no pudo efectuar la localización que se atribuye a la falta de precisión de palabra, ya que este método está programado considerando una precisión de palabra de 16 bits, los cuales son insuficientes para la presión necesaria.

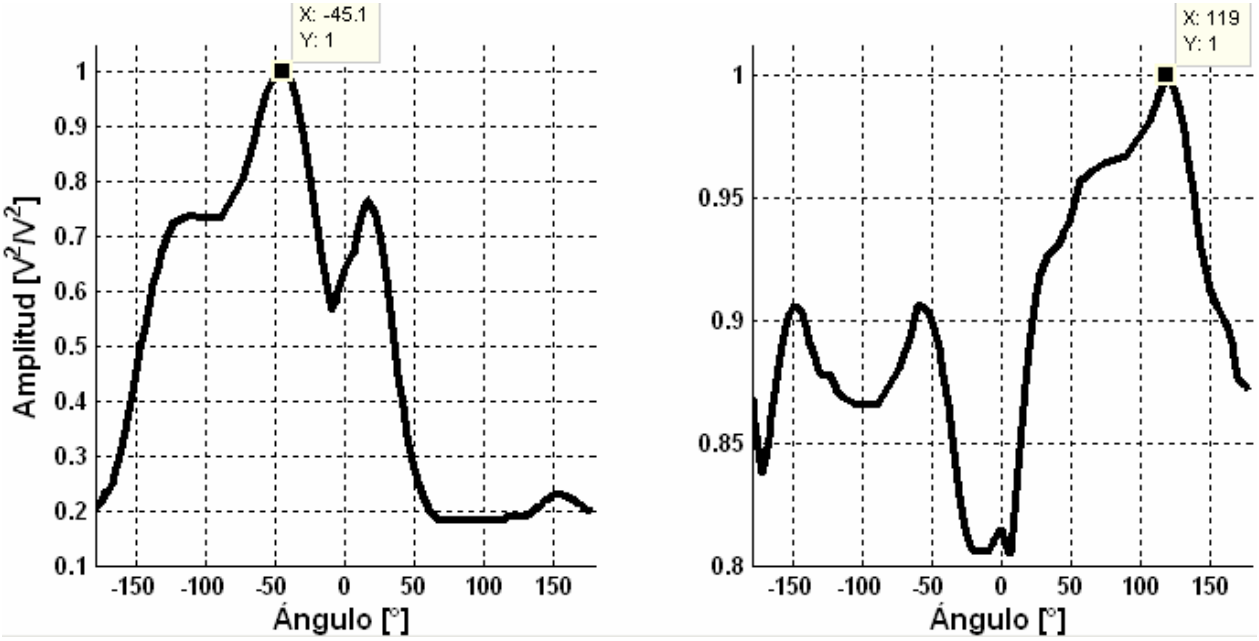
En las figuras 5.23, 5.24 y 5.25 se muestra el mapeo y estimación de ángulo con señales reales con el método GCC sumado. En el caso de la Figura 5.23.a) se hace incidir una señal que se ubica a cero grados del arreglo y en su gráfica se aprecia que la estimación se hace en los 0° que le corresponde. Para el caso de la Figura 5.23 b) la fuente de voz se ubica a 40° del arreglo y se observa que el valor estimado es de 41.81° existiendo un error de 1.81°.



a) **b)**

Figura 5.23. Mapeo y estimación de ángulo 0° en a) y en b) 41.81° con método GCC sumado.

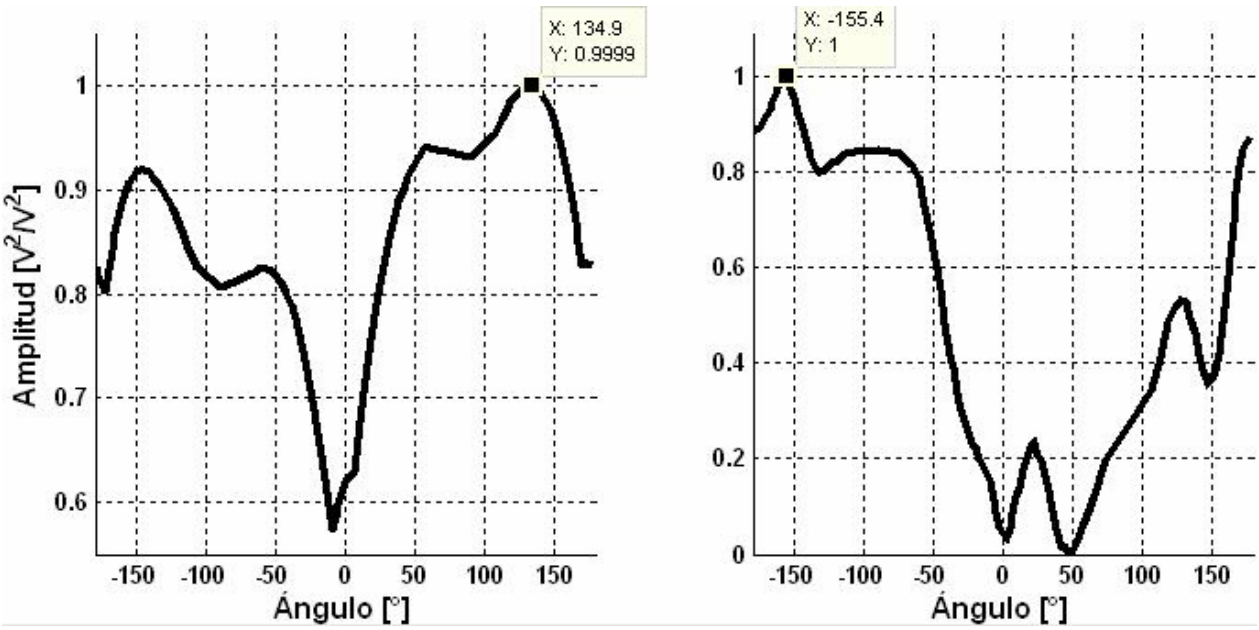
Otros ángulos a los que se probó el método GCC fueron - 50°, 110°, en la Figura 5.24.a) se muestra que para -50° se detecto - 45.1° por lo que hay un error de 4.9°, y en la Figura 5.24.b) para los 110° se estimó 119°, existiendo 9° de error.



a) **b)**

Figura 5.24. Mapeo y estimación de ángulo -45.1° en la gráfica del lado izquierdo y a la derecha 119° con método GCC sumado.

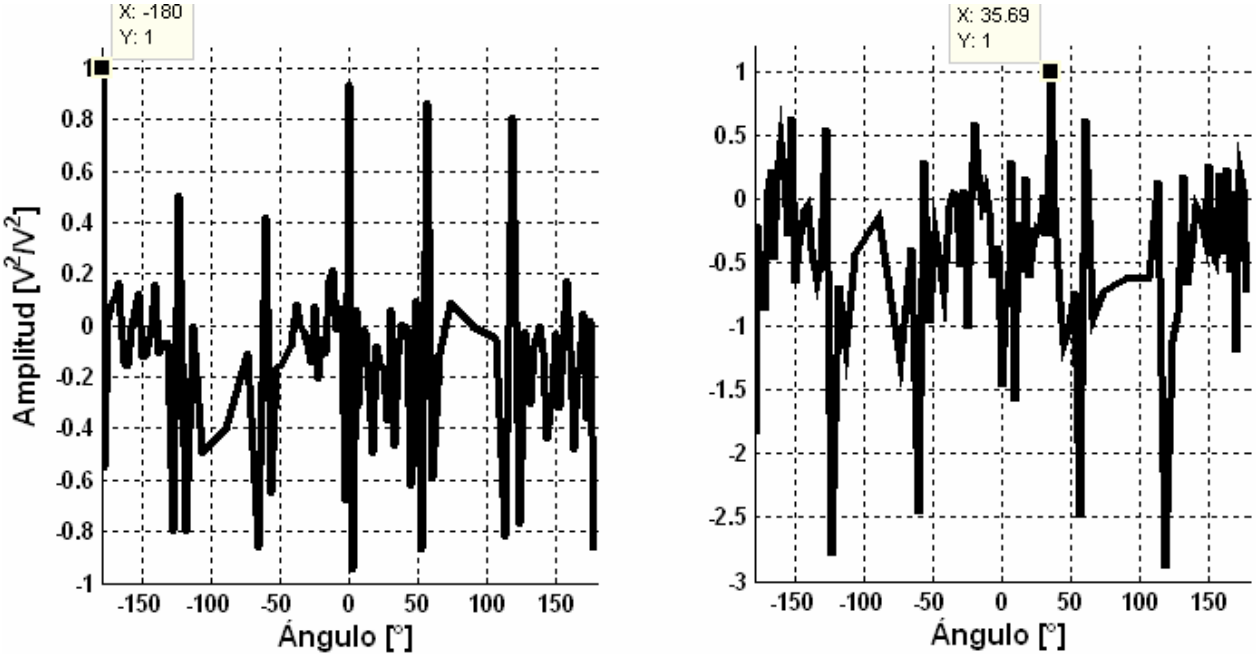
Para terminar con GCC en la Figura 5.26.a) incide a 130° y se estima a 134.9° con un error de 4.9°, y para 5.26.b) incide a -160° y se estima a -155.4° por lo que existe un error de 4.6°. Todo esto con señales reales.



a) **b)**

Figura 5.25. Mapeo y estimación de ángulo -45.1° en a) y en b) -155.4° con método GCC sumado.

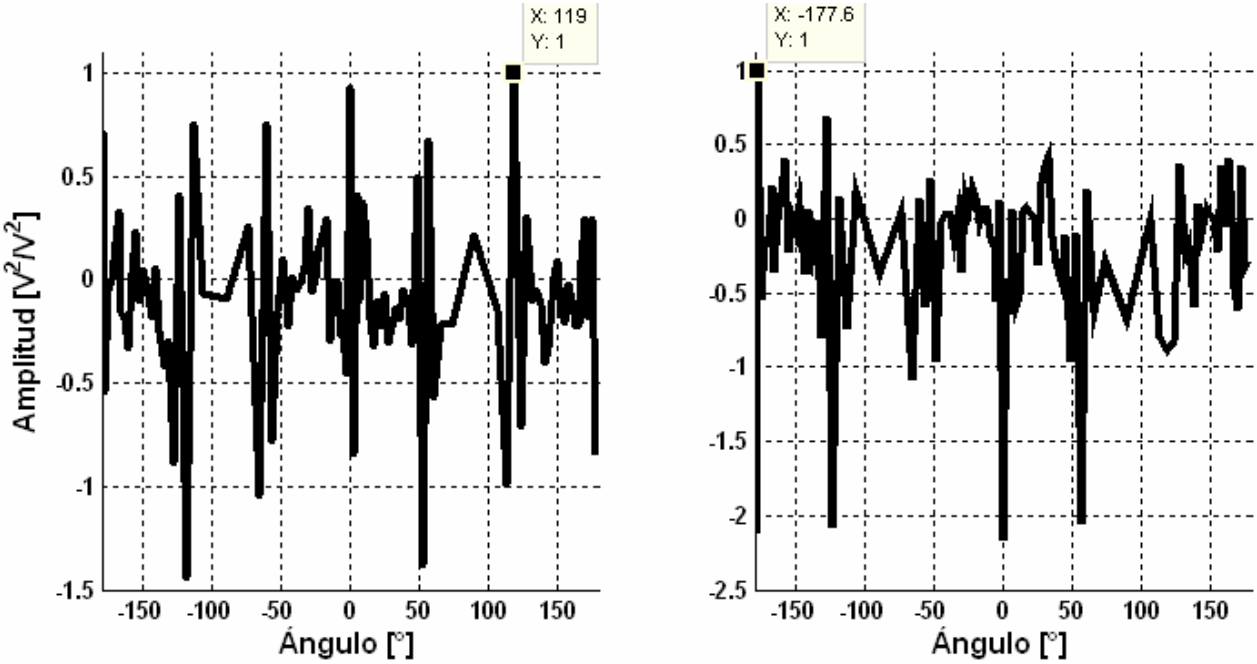
En las Figuras 5.27 y 5.28 se muestra los mapeos y estimación con el método transformada PHAT al igual que las anteriores con señales reales. En la Figura 5.26.a) se hizo incidir sobre el arreglo la señal de voz a 0°, se estima el máximo en -180°, para 5.26.b) la señal incidió a 40° detectándose 35.69°, por lo que existe un error de 4.31°.



a) **b)**

Figura 5.26. Mapeo y estimación de ángulo -180° en a) y en b) -35.69° con método transformada PHAT.

Para 5.27.a) incide a 110° estimándose 119° por lo que existe un error de 9° y 5.27.b) incide a -160° estimándose -177, existiendo un error de 17°, un error considerable en comparación a los otros casos.



a) **b)**

Figura 5.27. Mapeo y estimación de ángulo -110° en a) y en b) 160° con método transformada PHAT.

Se observa que cuando incide a 0° se estima en -180° debido al cono de confusión, aunque se observa en 5.26.a) que el segundo valor máximo se encuentra en 0°, es decir con este método es más fácil que se presenten estos errores de estimación debido al fenómeno de cono de confusión en comparación del método GCC. Además de que la presición de palabra afecta su desempeño ya que los 16 bits a los que está programado no son suficientes considerando que se remueve la amplitud de acuerdo a la transformada PHAT y solo se trabaja con la fase la cual es sensible al tamaño de palabra.

5.9. TIEMPO DE PROCESAMIENTO

Con la herramienta profile clock de CCS es posible saber aproximadamente el número de ciclos de reloj que tarda en realizar cada proceso, para GCC sumada, todo se efectúa en 878 052 ciclos de reloj, considerando que el DSP es de 1 GHz frecuencia de trabajo, el tiempo aproximado para efectuar el proceso es de 0.878 ms, lo cual es un tiempo mucho menor a los 2.7 ms que tarda en obtener las 128 muestras de ventana. El proceso más extensivo es el cálculo de la correlación cruzada.

Para el caso de la transformada de fase PHAT el proceso tarda 423 619 ciclos de reloj en efectuarlo, lo cual en tiempo son 0.423 ms, que es inferior a los 5.3 ms que tarda en tomarse las muestras de una ventana de 256 muestras a 48 kHz. Debido a que se elevó al cuadrado todo el proceso es aun más rápido que el GCC a pesar de ser más complicado por las transformadas y antitransformadas de Fourier que se aplican.

5.10. SÍNTESIS

Se aplicaron los algoritmos planteados en este trabajo a las señales de los micrófonos L, R y B, todo funcionando en el DSP TMS320C6416, ambos algoritmos pueden hacer la localización de una fuente de voz de acuerdo a las gráficas. Se muestra el proceso de ambos algoritmos con señales reales.

El método GCC sumado puede efectuar una mejor estimación del ángulo, ya que no se ve afectado por la precisión de palabra, a diferencia de transformada PHAT que al programarla a 16 bits no es suficiente, además de que GCC es menos sensible al fenómeno de cono de confusión a comparación de la transformada PHAT.

-Página en blanco intencionalmente-

6

CONCLUSIONES

Se implementaron y probaron los métodos actuales para realizar la localización de fuentes de voz con un número reducido de micrófonos. Estos métodos permiten conocer los distintos enfoques con los que se intenta resolver el problema de la localización de una fuente de voz.

De los métodos mencionados se eligió probar el GCC sumado y transformada de fase PHAT debido a que la construcción del arreglo es sencilla, no requiere de muchos micrófonos, en teoría puede realizar la localización de -180° a 180°, es decir, un giro completo al rededor del espacio considerado. Además, dichos métodos pueden obtener un estimado de la dirección en un lapso de tiempo muy corto, que pueden ayudar a hacer una localización más precisa que otros métodos más robustos en cuanto a proceso.

El GCC sumado puede hacer la estimación de ángulo al que se encuentra la fuente en 2.7 ms y cuando se utiliza la transformada de fase PHAT en aproximadamente 5.3 ms. Sin embargo, el procesamiento para ambos métodos se efectúa en tiempos menores a esto, siendo de 0.878 ms para GCC sumado y de 0.423 ms para transformada de fase PHAT. Por lo que es posible incluir otros procesos sin verse comprometidos los ya mencionados.

Los problemas que se tienen con ambos métodos es que son sensibles a ruidos externos tales como el sonido de motores, por lo que es recomendable su uso en un ambiente con pocas fuentes de ruido. La señal de voz no se puede apreciar al momento de aparecer uno de estos ruidos.

Como se observó en el capítulo 5, la detección con el método GCC solo se puede efectuar en los intervalos de -180° a -115°, -65° a 70° y de 120° a 175°, siendo imposible para los intervalos -70° a -105° y de 75° a 115° efectuar la estimación. El método PHAT, aunque se mostraron gráficas donde efectúa la estimación en la mayor parte de los casos, no puede obtener un estimado de la dirección de la fuente, lo cual se atribuye a que el método está programado con una precisión de palabra de 16 bits, que al momento de eliminar la componente de la amplitud, no la suficiente precisión para conservar la fase con la cual se trabaja.

Para este proyecto se requiere tener un detector de inicio y fin de palabra, ya que una fuente de voz en la realidad no es constante su presencia, por lo que es uno de los puntos en los que se puede seguir trabajando sobre el mismo proyecto.

Además, se requiere que la adquisición de muestras se obtenga por el DSP TMS320F2812 y se envíen al DSP TMS320C6416 para su procesamiento directamente sin necesidad de utilizar archivos, lo cual se podría realizar por comunicación serial mediante el periférico McBSP con el que cuentan ambos DSPs.

En teoría, para tener una mejor precisión en la estimación, se podría dividir el espacio en un mayor número de sub arreglos de pares de micrófonos, que implica utilizar un mayor número de micrófonos así como de un número mayor de datos a procesar.

Ya por último, se puede obtener un estimado con el método GCC sumado en un ambiente con pocas fuentes de ruido, a diferencia del método PHAT el cual requiere una mayor precisión de palabra, lo cual implica un convertidor analógico digital de mas bits de palabra así como un mayor tiempo de procesamiento.

BIBLIOGRAFÍA

[1] ALLEN, Ghavami**.** *Adaptive array systems. Fundamentals and applications*. Wiley Interscience. USA 2005.

[2] BALLOU, Glen. *Handbook for sound engineers*. Sams. USA 1991.

[3] BENESTY, SONDHI y HUANG. *Handbook of speech processing*. Springer. USA 2008.

[4] BENESTY, CHEN y HUANG. *Microphone array signal processing*. Springer. USA 2008.

[5] BRUTTI, OMOLOGO y SVAIZER. *Comparison between different sound source localization techniques based on a real data collection.* IEEE Italy 2008.

[6] BYOUNGHO, YOUNGJIN y YOUN-SIK. *Sound source localization for robot auditory system using the summed GCC method*. International conference on control automation and systems. Korea 2008.

[7] HAYES, Monson. *Statistical digital signal processing and modeling.* John Wiley & Sons. USA 1997.

[8] JOSEFSSON y PERSSON. *Conformal array antenna theory and design*. IEEE. USA 2006.

[9] KWON, PARK y PARK. *Sound source localization for robot auditory using the summed gcc method*. International Conference on Control, Automation and Systems. Seoul, Korea 2008.

[10] NAIDU, Prabhakar. *Sensor array signal processing*. CRC Press. USA 2001.

[11] OPPENHEIM, Alan. *Discrete Time Signal Processing.* Prentice Hall. USA 1999.

[12] OPPENHEIM, Alan. *Signal and systems*. Prentice Hall. USA 1987.

[13] PEEBLES, Payton. *Probability, random variables, and random signal principles*. Mc. Graw Hill. USA 1987.

[14] QIN, FU y YAN. *Subsample time delay estimation via improved GCC PHAT Algorithm.* IEEE. China 2008.

[15] SINCLAIR, Ian. *Audio and hi-fi handbook*. Newnes. Great Britain 1998.

[16] STRUMILLO, Pawel. *Advances in sound localization*. InTech. India 2011.

[17] VISSER, Hubergt. *Array and phased array antenna basics*. John Wiley & Sons, Ltd. UK. 2005.

[18] YONG-EUN, DONG-HYUN y CHANG-HA. *Sound source localization method using region selection*. Chonbuk National University. Korea 2011

GLOSARIO DE TÉRMINOS

Acústica.
Parte de la física que estudia las propiedades, producción, reproducción y propagación del sonido.

CCC.
Clasic Cross Correlation.

DFT.
Discrete Fourier Transform.

Eigenvector.
Vector no nulo que al multiplicarse por una matriz nxn dan por resultado un múltiplo escalar de si mismo, este escalar es el llamado eigenvalor.

FFT.
Fast Fourier Transform.

GCC.
Generalized Cross Correlation.

HRTF.
Head Related Transfer Function.

IPHAT.
Improved Phase Transform.

JTAG.
Joint Test Action Group.

LMS.
Least Mean Square.

PHAT.
Phase Transform.

SCOT.
Smoothed Coherente Transform.

SHRTF.
Sphere Head Related Transfer Function.

SSL.
Sound Source Localization.

TDE.
Time Difference Estimation.

TDOA.
Time Difference of Arrival.

Tiempo de Reverberación.
Tiempo que tarda en decaer gradualmente un sonido después de su emisión.

Apéndice A

Programas en C

En la Figura A.1 se muestra el proyecto con los archivos adjuntos para la el método GCC

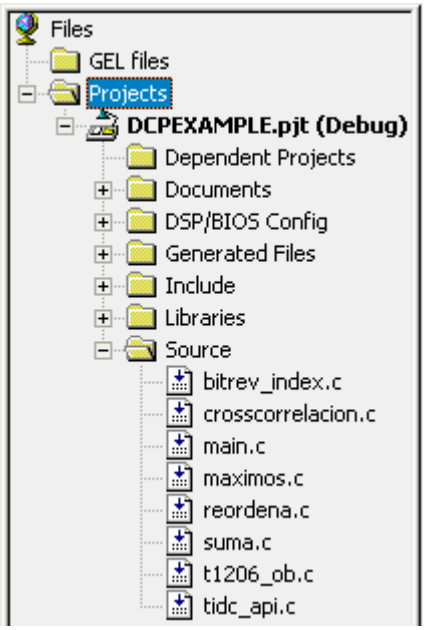


Figura A.1. Proyecto de Code Composer correspondiente al método GCC.

A continuación se muestra el código correspondiente para el método GCC.

```
#include "t1206_fn.h"
#include "tidc_api.h"

#include <csl.h>

#include <csl_irq.h>
#include <csl_timer.h>

#include "DSP_bitrev_cplx.h"

#define BUFF_SZ (0124)

long Dato = 0, *LAP;

unsigned short Buffer[BUFF_SZ];

////////////////////////////////////////
VARIABLES ALGORITMO

short R1[128], L1[128], B1[128], R2[128],
L2[128], B2[128], lleno, cont;
```

```
short *apR1, *apL1, *apB1, *apR2, *apL2,
*apB2, *auxR1, *auxL1, *auxB1, *auxR2,
*auxL2, *auxB2, *aux;

float angulos[96] = {
 -178.9550, -173.5565, -166.5978, -150.0000,
-133.4022, -126.4435, -121.0450, -116.4427,

 -112.3415, -108.5904, -105.0995, -101.8103,
-98.6822,  -95.6853,  -92.7972,  -90.0000,

  -87.2796,  -84.6243,  -82.0245,  -79.4712,
-76.9578,  -74.4775,  -72.0247,  -69.5941,

  -67.1808,  -64.7802,  -62.3880,  -60.0000,
-57.6120,  -55.2198,  -52.8192,  -50.4059,

  -47.9753,  -45.5225,  -43.0422,  -40.5288,
-37.9755,  -35.3757,  -32.7204,  -30.0000,

  -27.2028,  -24.3147,  -21.3178,  -18.1897,
-14.9005,  -11.4096,   -7.6585,   -3.5573,

    1.0450,    6.4435,   13.4022,   30.0000,
46.5978,   53.5565,   58.9550,   63.5573,

   67.6585,   71.4096,   74.9005,   78.1897,
81.3178,   84.3147,   87.2028,   90.0000,

   92.7204,   95.3757,   97.9755,  100.5288,
103.0422,  105.5225,  107.9753,  110.4059,

  112.8192,  115.2198,  117.6120,  120.0000,
122.3880,  124.7802,  127.1808,  129.5941,

  132.0247,  134.4775,  136.9578,  139.4712,
142.0245,  144.6243,  147.2796,  150.0000,

  152.7972,  155.6853,  158.6822,  161.8103,
165.0995,  168.5904,  172.3415,  176.4427},
ESTIMADO, *PS, p;

int ex;

////////////////////////////////////
void intTimer(void);

 int x[] =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,1,2,3
,4,5,6,7,8,9,10,11,12,13,14,15,16};

 short x1[] =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,1,2,3
,4,5,6,7,8,9,10,11,12,13,14,15,16};

 short y[] =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0}, apy;

 short W[] = {16384, 15137, 11585, 6270, 0, -
6270, -11585, -15137, 0, -12540, -23170, -
30274, -32768, -30274, -12540, -23170};

 int LR[49], RB[49], BL[49], *RES, LR1[96],
RB1[96], BL1[96];

 int  valores[4] = {0, 0, 0, 0}, *val;

IRQ_Config tConfig = { //CONFIGURACIÓN DE
INTERRUPCION TIMER 0
intTimer,
0x00000000,
IRQ_CCMASK_DEFAULT,
IRQ_IEMASK_ALL,
};

TIMER_Config miConfig = { // CONFIGURACIÓN DE
TIMER 0

0x00000385,/* ctl */
0x00000516,/* prd */
0x00000000/* cnt */

};

TIMER_Handle miTimer; // MANEJADOR DE TIMER 0

void crosscorrelacion(short *ap1, short *ap2,
short tam, short sam, int *res);

void reordena(int *v1, int *v2, int *v3, int
*d1, int *d2, int *d3);

void suma(int*, int*, int*);

void maximos(int *P, short tam, int *valPos);

void main(void){

//////// CÓDIGO

cont = 0;
lleno = 0;


apR1 = R1;
apL1 = L1;
apB1 = B1;

apR2 = R2;
apL2 = L2;
apB2 = B2;

auxR1 = R1;
auxL1 = L1;
auxB1 = B1;

auxR2 = R2;
auxL2 = L2;
auxB2 = B2;

PS = angulos;

val = valores;

val++;

///////

CSL_init();

miTimer = TIMER_open(TIMER_DEV0,
TIMER_OPEN_RESET); // ABRE PUERTO TIMER 0

TIMER_config(miTimer, &miConfig); //
CONFIGURA TIMER 0 ACORDE A LA ESTRUCTURA
CONFIGURACIÓN TIMER
```

```
IRQ_globalDisable(); // DESHABILITA
INTERRUPCIONES DE FORMA GLOBAL

IRQ_clear(IRQ_EVT_TINT0); // LIMPIA FLAG
REGISTER CORRESPONDIENTE A INTERRUPCION
TIMER0

IRQ_enable(IRQ_EVT_TINT0); // HABILITA
INTERRUPCION TIMER 0 EN IER

IRQ_config(IRQ_EVT_TINT0, &tConfig); //
CONFIGURA INTERRUPCIÓN CORRESPONDIENTE A
TIMER 0 ACORDE A ESTRUCTURA CONFIGURACION

TIMER_start(miTimer);//INICIA TIMER

LAP = &Dato;

dc_configure(&Ths1206_1);//

IRQ_globalEnable();//HABILITA INTERRUPCIONES

dc_readblock(&Ths1206_1, Buffer, BUFF_SZ, 0);

dc_readsample(&Ths1206_1, LAP);

//TIMER_close(miTimer);

DSP_radix2(16, x1, W);

bitrev_index(y, 16);

DSP_bitrev_cplx((int*) x1, y, 16);

DSP_bitrev_cplx( x, y, 16);

DSP_bitrev_cplx( x+16, y, 16);

DSP_bitrev_cplx( x, y, 16);

while(1){

if( lleno != 0 ){

apR1 = auxR2; //apR1 apunta al vector que se
estaba llenando y se cambia de forma que se
llene el otro buffer

apL1 = auxL2;// SE APUNTA AL INICIO DE CADA
BUFFER

apB1 = auxB2;

apR2 = auxR1;

apL2 = auxL1;

apB2 = auxB1;

aux = auxR2; // SE CONMUTAN APUNTADORES ENTRE
BUFFER A LOS QUE APUNTAN

auxR2 = auxR1;

auxR1 = aux;

aux = auxL2;

auxL2 = auxL1;

auxL1 = aux;

aux = auxB2;

auxB2 = auxB1;

auxB1 = aux;

//////////////////////////////////////////
PROCESO cross correlacion de
señales////////////////

RES = LR;

crosscorrelacion(apL2, apR2, 128, 24, RES);

RES = RB;

crosscorrelacion(apR2, apB2, 128, 24, RES);

RES = BL;

crosscorrelacion(apB2, apL2, 128, 24, RES);

reordena( LR, RB, BL, LR1, RB1, BL1);

suma(LR1, RB1, BL1);//SUMA SE GUARDA EN LR1

maximos(LR1, 96, valores);

ESTIMADO = *(PS + *val - 1);

//////////////////////////////////////////
//////////////////////////////////////////
///////

lleno = 0;

}

};

}

void intTimer(void)
{

cont++;

*(apR1++) = 0x20ff;
*(apL1++) = 0x20ff;
*(apB1++) = 0x20ff;

if(cont == 127){

cont = 0;

lleno = 1;

}

IRQ_clear(IRQ_EVT_TINT0);

dc_readsample(&Ths1206_1, LAP);

}

/* ap1 apunta a vector 1, ap2  apunta a
vector 2, tam = tamaño del vector, sam numero
```

```
de  muestras de funcion crosscorrelacion,
*res apunta a vector resultado*/

void crosscorrelacion(short *ap1, short *ap2,
short tam, short sam, int *res)/*CALCULA
CROSSCORRELACIÓN PERO SOLO UNA PARTE*/
{
short con1 = 1;

short con2 = sam;

int ACC = 0, ACX;

short *aux1, *aux2;

aux1 = ap1;

aux2 = ap2;

while (con2 >= 0){

       ap2 = aux2 + con2;

       while (con1 <= tam - con2){

              ACX = ((int )(*(ap1++)))*((int
)(*(ap2++)));

              ACC = ACC + (ACX >> 11);

              con1++;

       }

       con1 = 1;

       ap1 = aux1;

       *(res++) = ACC;

       ACC = 0;

       con2--;

}

con2 = 1;

ap2 = aux2;

while (con2 <= sam){

       ap1 = aux1 + con2;

       while (con1 <= tam - con2){

       ACX = ((int )(*(ap1++)))*((int
)(*(ap2++)));

       ACC = ACC + (ACX >> 11);

       con1++;

       }
con1 = 1;

ap2 = aux2;

*(res++) = ACC;

ACC = 0;

con2++;

}

return;
}


void reordena(int *v1, int *v2, int *v3, int
*d1, int *d2, int *d3){ // funcion especial
para reordenar datos

short cont = 24;

v1 = v1 + 24;

v2 = v2 + 24;

v3 = v3 + 24;

while(cont >= 1)
{

*(d1++) = *(v1--);

*(d2++) = *(v2--);

*(d3++) = *(v3--);

cont--;

}

while (cont <= 47)
{

*(d1++) = *(v1++);

*(d2++) = *(v2++);

*(d3++) = *(v3++);

cont++;
}

while (cont >= 25)
{

*(d1++) = *(v1--);

*(d2++) = *(v2--);

*(d3++) = *(v3--);

cont--;
}
return;
}

void suma( int *LR, int *RB, int *BL ){

int aux, cont = 1, *ORB, *OBL;

ORB = RB;

OBL = BL;

RB = RB + 69;
```

```
BL = BL + 28;

while(cont <= 26)
{

aux = *LR + *(RB++) + *(BL++);

*(LR++) = aux;

cont++;

}

RB = ORB;

while(cont <= 68)
{

aux = *LR + *(RB++) + *(BL++);

*(LR++) = aux;

cont++;

}

BL = OBL;

while(cont <= 96)
{

aux = *LR + *(RB++) + *(BL++);

*(LR++) = aux;

cont++;

}

return;

}

void maximos(int *P, short tam, int *valPos){
// P es el vector de datos, tam tamaño del
vector, vector que apunta a valor y posicion
// valpos apunta a arreglo de 4 datos, la
primera posición indica el valor maximo,
// la segunda la posicion dentro del arreglo
del valor maximo, tercer posicion valor del
segundo máximo,
// cuarta posicion posicion del segundo
maximo

short cont = 2;

*valPos = *(P++); //VALOR PIVOTE

*(valPos+1) = 1;

while( cont <= tam ){

        if(*valPos < *P ){

        *(valPos+2) = *valPos;

        *(valPos+3) = *(valPos+1);

        *valPos = *P;

        *(valPos+1) = cont;

        }

        P++;

        cont++;

        }

}
/*
============================================
========================== */
/*
*/
/*  TEXAS INSTRUMENTS, INC.
*/
/*
*/
/*  NAME
*/
/*      bitrev_index
*/
/*
*/
/*  USAGE
*/
/*      This function has the prototype:
*/
/*
*/
/*      void bitrev_index(short *index, int
n);                              */
/*
*/
/*      index[sqrt(n)] : Pointer to index
table that is returned by the     */
/*                       routine.
*/
/*      n              : Number of complex
array elements to bit-reverse.   */
/*
*/
/*  DESCRIPTION
*/
/*      This routine calculates the index
table for the DSPLIB function     */
/*      bitrev_cplx which performs a complex
bit reversal of an array of    */
/*      length n. The length of the index
table is 2^(2*ceil(k/2)) where    */
/*      n = 2^k. In other words, the length
of the index table is sqrt(n)   */
/*      for even powers of radix.
*/
/*
*/
/*
============================================
========================== */

void bitrev_index(short *index, int n)
{
    int   i, j, k, radix = 2;
    short nbits, nbot, ntop, ndiff, n2,
raddiv2;

    nbits = 0;
```

```
    i = n;
    while (i > 1)
    {
        i = i >> 1;
        nbits++;
    }

    raddiv2 = radix >> 1;
    nbot    = nbits >> raddiv2;
    nbot    = nbot << raddiv2 - 1;
    ndiff   = nbits & raddiv2;
    ntop    = nbot + ndiff;
    n2      = 1 << ntop;
```

```
    index[0] = 0;
    for ( i = 1, j = n2/radix + 1; i < n2 -
1; i++)
    {
        index[i] = j - 1;

        for (k = n2/radix; k*(radix-1) < j; k
/= radix)
            j -= k*(radix-1);

        j += k;
    }
    index[n2 - 1] = n2 - 1;
}
```

En la Figura A.2 se muestra el proyecto de Code Composer correspondiente al método de Transformada PHAT donde se observan los archivos adjuntos.

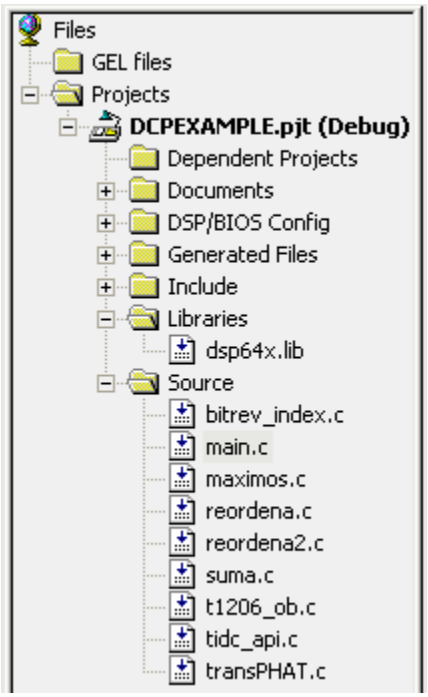


Figura A.2. Proyecto de Code Composer correspondiente al método Transformada PHAT.

A continuación se muestra el código correspondiente al programa del método de la transformada PHAT junto con sus funciones.

```
#include "t1206_fn.h"
#include "tidc_api.h"

#include <csl.h>
#include <math.h>
#include <csl_irq.h>
#include <csl_timer.h>

#include "DSP_bitrev_cplx.h"

#define BUFF_SZ (0124)

long Dato = 0, *LAP;
```

```
unsigned short Buffer[BUFF_SZ];

////////////////////////////////////////
VARIABLES ALGORITMO

float angulos[96] = {

 -178.9550, -173.5565, -166.5978, -150.0000,
-133.4022, -126.4435, -121.0450, -116.4427,

 -112.3415, -108.5904, -105.0995, -101.8103,
-98.6822,  -95.6853,  -92.7972,  -90.0000,
```

```
  -87.2796,  -84.6243,  -82.0245,  -79.4712,
-76.9578,  -74.4775,  -72.0247,  -69.5941,

  -67.1808,  -64.7802,  -62.3880,  -60.0000,
-57.6120,  -55.2198,  -52.8192,  -50.4059,

  -47.9753,  -45.5225,  -43.0422,  -40.5288,
-37.9755,  -35.3757,  -32.7204,  -30.0000,

  -27.2028,  -24.3147,  -21.3178,  -18.1897,
-14.9005,  -11.4096,   -7.6585,   -3.5573,

    1.0450,    6.4435,   13.4022,   30.0000,
46.5978,   53.5565,   58.9550,   63.5573,

   67.6585,   71.4096,   74.9005,   78.1897,
81.3178,   84.3147,   87.2028,   90.0000,

   92.7204,   95.3757,   97.9755,  100.5288,
103.0422,  105.5225,  107.9753,  110.4059,

  112.8192,  115.2198,  117.6120,  120.0000,
122.3880,  124.7802,  127.1808,  129.5941,

  132.0247,  134.4775,  136.9578,  139.4712,
142.0245,  144.6243,  147.2796,  150.0000,

  152.7972,  155.6853,  158.6822,  161.8103,
165.0995,  168.5904,  172.3415,  176.4427},
ESTIMADO, *PS, p, ESTIMADO1;

//////////////////////////////////////

 short R1[512], L1[512], B1[512], R2[512],
L2[512], B2[512], LR[512], RB[512], BL[512],
lr[49], rb[49], bl[49], lr1[96],  rb1[96],
bl1[96];

 short maxval[4] = { 0, 0, 0, 0};

 short y[256], lleno, cont;

 short *apR1, *apL1, *apB1, *apR2, *apL2,
*apB2, *auxR1, *auxL1, *auxB1, *auxR2,
*auxL2, *auxB2, *aux;

 short W[] ={  32767,      0,   32757,   -
804,  32728,  -1608,  32678,  -2410,  32609,
-3212,  32521, -4011,

       32412,  -4808,  32285,   -5602,
32137,  -6393,  31971,  -7179,  31785,  -
7962,  31580,  -8739,

       31356,  -9512,  31113,  -10278,
30852, -11039,  30571, -11793,  30273, -
12539,  29956, -13279,

       29621, -14010,  29268,  -14732,
28898, -15446,  28510, -16151,  28105, -
16846,  27683, -17530,

       27245, -18204,  26790,  -18868,
26319, -19519,  25832, -20159,  25329, -
20787,  24811, -21403,

       24279, -22005,  23731,  -22594,
23170, -23170,  22594, -23731,  22005, -
24279,  21403, -24811,

       20787, -25329,  20159,  -25832,
19519, -26319,  18868, -26790,  18204, -
27245,  17530, -27683,

       16846, -28105,  16151,  -28510,
15446, -28898,  14732, -29268,  14010, -
29621,  13279, -29956,

       12539, -30273,  11793,  -30571,
11039, -30852,  10278, -31113,   9512, -
31356,   8739, -31580,

        7962, -31785,   7179,  -31971,
6393, -32137,   5602, -32285,   4808, -32412,
4011, -32521,

        3212, -32609,   2410,  -32678,
1608, -32728,    804, -32757,      0, -32767,
-804, -32757,

        1608, -32728,  -2410,  -32678,  -
3212, -32609,  -4011, -32521,  -4808, -32412,
-5602, -32285,

       -6393, -32137,  -7179,  -31971,  -
7962, -31785,  -8739, -31580,  -9512, -31356,
-10278, -31113,

      -11039, -30852, -11793,  -30571, -
12539, -30273, -13279, -29956, -14010, -
29621, -14732, -29268,

      -15446, -28898, -16151,  -28510, -
16846, -28105, -17530, -27683, -18204, -
27245, -18868, -26790,

      -19519, -26319, -20159,  -25832, -
20787, -25329, -21403, -24811, -22005, -
24279, -22594, -23731,

      -23170, -23170, -23731,  -22594, -
24279, -22005, -24811, -21403, -25329, -
20787, -25832, -20159,

      -26319, -19519, -26790,  -18868, -
27245, -18204, -27683, -17530, -28105, -
16846, -28510, -16151,

      -28898, -15446, -29268,  -14732, -
29621, -14010, -29956, -13279, -30273, -
12539, -30571, -11793,

      -30852, -11039, -31113,  -10278, -
31356, - 9512, -31580, - 8739, -31785, -7962,
-31971,  -7179,

      -32137,  -6393, -32285,   -5602, -
32412,  -4808, -32521,  -4011, -32609, -3212,
-32678,  -2410,

      -32728,  -1608, -32757,    -804};

short x1[] = {1, 1, 2, 2, 3, 3, 4, 4, 5, 5,
6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11, 11, 12,
12, 13, 13, 14, 14, 15, 15, 16, 16};

void intTimer(void);
void reordena(short *v1, short *v2, short
*v3, short *d1, short *d2, short *d3,short
tam);
```

```
void reordena2(short *v1, short *v2, short
*v3, short *d1, short *d2, short *d3);
void suma( short *LR, short *RB, short *BL );
void maximos(short *P, short tam, short
*valPos);
void reordena(short *v1, short *v2, short
*v3, short *d1, short *d2, short *d3, short
tam);
void suma(short*, short*, short*);
void maximos(short *P, short tam, short
*valPos);
void transPHAT(short *v1, short *v2,short
tam, short *res);

IRQ_Config tConfig = { //CONFIGURACION DE
INTERRUPCION TIMER 0

intTimer,
0x00000000,
IRQ_CCMASK_DEFAULT,
IRQ_IEMASK_ALL,
};

TIMER_Config miConfig = { // CONFIGURACION DE
TIMER 0

0x00000385,/* ctl */
0x00000516,/* prd */
0x00000000/* cnt */

};

TIMER_Handle miTimer; // MANEJADOR DE TIMER 0

void main(void){

//////// CÓDIGO

cont = 0;
lleno = 0;

apR1 = R1;
apL1 = L1;
apB1 = B1;

apR2 = R2;
apL2 = L2;
apB2 = B2;

auxR1 = R1;
auxL1 = L1;
auxB1 = B1;

auxR2 = R2;
auxL2 = L2;
auxB2 = B2;

PS = angulos;

///////

CSL_init();

miTimer = TIMER_open(TIMER_DEV0,
TIMER_OPEN_RESET); // ABRE PUERTO TIMER 0

TIMER_config(miTimer, &miConfig); //
CONFIGURA TIMER 0 ACORDE A LA ESTRUCTURA
CONFIGURACIÓN TIMER

IRQ_globalDisable(); // DESHABILITA
INTERRUPCIONES DE FORMA GLOBAL

IRQ_clear(IRQ_EVT_TINT0); // LIMPIA FLAG
REGISTER CORRESPONDIENTE A INTERRUPCION
TIMER0

IRQ_enable(IRQ_EVT_TINT0); // HABILITA
INTERRUPCION TIMER 0 EN IER

IRQ_config(IRQ_EVT_TINT0, &tConfig); //
CONFIGURA INTERRUPCIÓN CORRESPONDIENTE A
TIMER 0 ACORDE A ESTRUCTURA CONFIGURACION

TIMER_start(miTimer);//INICIA TIMER



dc_configure(&Ths1206_1);//

IRQ_globalEnable();//HABILITA INTERRUPCIONES

dc_readblock(&Ths1206_1, Buffer, BUFF_SZ, 0);

dc_readsample(&Ths1206_1, LAP);

//TIMER_close(miTimer);



bitrev_index(y, 256);

DSP_bitrev_cplx((int*) x1, y, 16);


while(1){

if( lleno != 0 ){

apR1 = auxR2; //apR1 apunta al vector que se
estaba llenando y se cambia de forma que se
llene el otro buffer

apL1 = auxL2;// SE APUNTA AL INICIO DE CADA
BUFFER

apB1 = auxB2;

apR2 = auxR1;

apL2 = auxL1;

apB2 = auxB1;

aux = auxR2; // SE CONMUTAN APUNTADORES ENTRE
BUFFER A LOS QUE APUNTAN

auxR2 = auxR1;

auxR1 = aux;

aux = auxL2;

auxL2 = auxL1;

auxL1 = aux;

aux = auxB2;

auxB2 = auxB1;
```

```
auxB1 = aux;

//////////////////////////////////////////
PROCESO cross correlación de
señales////////////////

DSP_radix2(256, apR2, W);

DSP_bitrev_cplx((int*) apR2, y, 256);

DSP_radix2(256, apL2, W);

DSP_bitrev_cplx((int*) apL2, y, 256);

DSP_radix2(256, apB2, W);

DSP_bitrev_cplx((int*) apB2, y, 256);

transPHAT(apL2, apR2, 256, LR);//transformada
phat devuelve resultado conjugado

transPHAT(apR2, apB2, 256, RB);

transPHAT(apB2, apL2, 256, BL);

DSP_radix2(256, LR, W);

DSP_bitrev_cplx((int*) LR, y, 256);
//antitransformada para valores mas exactos
obtener el conjugado y multiplicar por en
inverso de N = 256

DSP_radix2(256, RB, W);

DSP_bitrev_cplx((int*) RB, y, 256);

DSP_radix2(256, BL, W);

DSP_bitrev_cplx((int*) BL, y, 256);

reordena(LR, RB, BL, lr, rb, bl, 512);//se
forma vector real de 49 datos

reordena2( lr, rb, bl, lr1, rb1, bl1);//se
forman los tres vectores reales de 96 datos

suma(lr1, rb1, bl1);//suma los tres vectores
reales y se gurda el resultado en lr1

maximos( lr1, 96, maxval);

ESTIMADO = *(PS + maxval[1] -1);//valor
estimado

ESTIMADO1 = *(PS + maxval[3] -1);//valor
estimado

//////////////////////////////////////////
//////////////////////////////////////////
///////

lleno = 0;

}

};


}
```

```
void intTimer(void)
{

cont++;

*(apR1++) = 0x20ff;
*(apR1++) = 0x0000;
*(apL1++) = 0x20ff;
*(apL1++) = 0x0000;
*(apB1++) = 0x20ff;
*(apB1++) = 0x0000;

if(cont == 256){

cont = 0;

lleno = 1;

}


IRQ_clear(IRQ_EVT_TINT0);

dc_readsample(&Ths1206_1, LAP);
return;
}


void reordena(short *v1, short *v2, short
*v3, short *d1, short *d2, short *d3,short
tam){ // funcion especial para reordenar
datos

short cont = 1;

v1 = v1 + tam - 48;

v2 = v2 + tam - 48;

v3 = v3 + tam - 48;

while(cont <= 24 )
{

*(d1++) = *(v1++);

*(d2++) = *(v2++);

*(d3++) = *(v3++);

v1++;

v2++;

v3++;

cont++;

}

v1 = v1 - tam;

v2 = v2 - tam;

v3 = v3 - tam;


while (cont <= 49)
```

```
{

*(d1++) = *(v1++);

*(d2++) = *(v2++);

*(d3++) = *(v3++);

v1++;

v2++;

v3++;

cont++;
}


return;
}


void reordena2(short *v1, short *v2, short
*v3, short *d1, short *d2, short *d3){ //
función especial para reordenar datos

short cont = 1;

v1 = v1 + 24;

v2 = v2 + 24;

v3 = v3 + 24;

while(cont <= 24)
{
*(d1++) = *(v1--);

*(d2++) = *(v2--);

*(d3++) = *(v3--);

cont++;

}

while(cont <= 72)
{
*(d1++) = *(v1++);

*(d2++) = *(v2++);

*(d3++) = *(v3++);

cont++;

}

while(cont <= 96)
{
*(d1++) = *(v1--);

*(d2++) = *(v2--);

*(d3++) = *(v3--);

cont++;

}
 return;
}

void suma( short *LR, short *RB, short *BL ){

short aux, cont = 1, *ORB, *OBL;

ORB = RB;

OBL = BL;

RB = RB + 69;

BL = BL + 28;

while(cont <= 26)
{

aux = *LR + *(RB++) + *(BL++);

*(LR++) = aux;

cont++;

}

RB = ORB;

while(cont <= 68)
{

aux = *LR + *(RB++) + *(BL++);

*(LR++) = aux;

cont++;

}

BL = OBL;

while(cont <= 96)
{

aux = *LR + *(RB++) + *(BL++);

*(LR++) = aux;

cont++;

}

return;

}

void transPHAT(short *v1, short *v2,short
tam, short *res)
{
short ax, bx, ay, by, ax2, bx2, ay2, by2,
den, ter1, ter2, ter3, ter4, ter5, ter6,
ter7, pR, pI, cont = 1;

while (cont <= tam)
{

ax = *(v1++);
bx = *(v1++);

ay = *(v2++);
by = *(v2++);
```

```
        ax2 = ax * ax;
        ax2 = ax2 >> 11;

        ay2 = ay * ay;
        ay2 = ay2 >> 11;

        bx2 = bx * bx;
        bx2 = bx2 >> 11;

        by2 = by * by;
        by2 = by2 >> 11;

        ter1 = (ax2 * ay2) >> 11;

        ter2 = (bx2 * by2) >> 11;

        ter3 = (ax2 * by2) >> 11;

        ter4 = (ay2 * bx2) >> 11;

        ter5 = (ax * ay) >> 11;

        ter5 = ter5 * ( (bx * by)>>11 );

        ter5 = ter5 >> 11;

        ter5 = 4 * ter5;

        ter6 = ax2 - by2;

        ter6 = (ter6 * bx)>>11;

        ter6 = (ter6 * ax)>>11;

        ter6 = 2 * ter6;

        ter7 = bx2 - ax2;

        ter7 = (ter7 * by) >> 11;

        ter7 = (ter7 * ay) >> 11;

        ter7 = 2 * ter7;

         den = ter1 + ter2 + ter3 + ter4;

          pR = ter1 + ter2 - ter3 - ter4 +
ter5;

          pI =  ter6 + ter7;

*(res++) = pR / den;

*(res++) = -pI / den; //El conjugado
directamente

cont++;

}

return;
}
```

Para las captura de las señales se utiliza la tarjeta DSK TMS320F2812 con su convertidor analógico digital interno de 16 canales, a continuación se muestra el código de la función principal que programa la adquisición de muestras.

```
#include <DSP281x_Device.h>
#include <DSP281x_GlobalPrototypes.h>
#include <DSP281x_DefaultIsr.h>
#include <DSP281x_Examples.h>

interrupt void Timer0_interrup(void);
interrupt void adc_interrup(void);

void RETRA(void);

unsigned int *leeA1;

int SEN[3], con1 = 1, con2 = 1;
int *ap1;

signed int *canal1, *canal2, *canal3;


void main(void){

DisableDog();//deshabilita WD
asm("   SETC           OBJMODE");
InitPll(0xA);

EALLOW;
SysCtrlRegs.PLLCR.bit.DIV=0xA;// SYSCLK
=150MHz
SysCtrlRegs.HISPCP.bit.HSPCLK=0x0;//
HSPCLK=SYSCLK
SysCtrlRegs.PCLKCR.bit.ADCENCLK=1;
       //Habilita el reloj del ADC
InitCpuTimers();//INICIA CPU TIMERS
CpuTimer0Regs.TCR.bit.TSS=0x1;//CONTADOR
DETENIDO
CpuTimer0Regs.PRD.all=0x0C35;  //PRD =
300X10^6
CpuTimer0Regs.TIM.all=0x0C35;
CpuTimer0Regs.TCR.bit.FREE=0x0;//Configura
que el contador se detenga en la posicion en
que se encuentre
CpuTimer0Regs.TCR.bit.SOFT=0x0;
CpuTimer0Regs.TCR.bit.TIE=0x1; //habilita
interrupcion por contador
GpioMuxRegs.GPAMUX.all=0x0000;
GpioMuxRegs.GPADIR.all=0xFFFF;

GpioDataRegs.GPADAT.bit.GPIOA0=0;

InitAdc();

AdcRegs.ADCTRL3.bit.ADCEXTREF=0x0;
```

```
AdcRegs.ADCTRL1.bit.CPS=0x0;
       //Preescalador de reloj. Si es igual
a 0 Fclk=clk/1. Si es igual a 1 Fclk = clk/2

                     //75MHz conversion
AdcRegs.ADCTRL1.bit.SEQ_OVRD=0x0;
AdcRegs.ADCTRL3.bit.ADCCLKPS=0x03;   //0X06
Divisor de reloj ADCLK = HSPCLK/(ADCCLKPS+1)
75MHz/2=37.5MHz
AdcRegs.ADCTRL3.bit.SMODE_SEL=0x0;
AdcRegs.ADCTRL1.bit.SUSMOD=0x2;

              //Modo emulación-suspensión.
Bits que determinan que sucede cuando sucede
una suspension ocurre como por un brekpoint
AdcRegs.ADCTRL1.bit.ACQ_PS=0x1;
       //Ancho del pulso SOC periodo =
ADCTRL1[11:8]+1 veces el ADCLK
AdcRegs.ADCTRL1.bit.CONT_RUN=0x0;    //Modo
de conversión. Si es 1 la conversion continua
despues de alcanzar EOS, si es 0 la
conversion se detiene antes de alcanzar EOS
AdcRegs.ADCTRL1.bit.SEQ_CASC=0x1;    //Si es
0 SEQ1 y SEQ2 operan como dos secuenciadores
distintos, si es 1 SEQ1 y SEQ2 operan como un
unico SEQ de 16 bits y opera en modo cascada
AdcRegs.ADCMAXCONV.bit.MAX_CONV1=0x2;//Numero
maximo de conversiones por sesión
AdcRegs.ADCCHSELSEQ1.bit.CONV00=0;
AdcRegs.ADCCHSELSEQ1.bit.CONV01=1;
AdcRegs.ADCCHSELSEQ1.bit.CONV02=2;
AdcRegs.ADCTRL2.bit.RST_SEQ1=0x1;
AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;

AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1=0x1;//Habili
ta interrupción INT SEQ1
AdcRegs.ADCTRL2.bit.INT_MOD_SEQ1=0x0;

asm("  SETC          INTM");
//Deshabilita interrupciones
EALLOW;


PieCtrlRegs.PIEIER1.bit.INTx7=0x1;
//habilita interrupción INT1.7 en PIE
PieVectTable.TINT0=&Timer0_interrup;//Liga
rutina interrupcion con la tabla vector
PieCtrlRegs.PIEIER1.bit.INTx6=0x1;
       //Habilita interrupcion PIE INT1.6
PieVectTable.ADCINT=&adc_interrup;  //Asocia
el PIE tabla vector de interrupciones con la
función interrupción adc_interrup
PieCtrlRegs.PIECRTL.bit.ENPIE=0x1;
       //Habilita interrupciones PIE
IER            |=M_INT1;
              //Habilita INT1 en registro
IER
EINT;
              //Habilita interrupciones INTM
= 0
ERTM;
              //Habilita interrupciones en
modo debugger

CpuTimer0Regs.TCR.bit.TSS=0x0;//Inicia conteo


while(1){

}
}

interrupt void Timer0_interrup(void){
asm("  NOP");
AdcRegs.ADCTRL2.bit.SOC_SEQ1=0x1;
PieCtrlRegs.PIEACK.bit.ACK1=0x1;//Cambia
estado de PIEIFR asociado a INT1
return;
}

interrupt void adc_interrup(void){

AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1;

if (con1 == 1){

canal1 = (signed int*)0x8000;//
canal2 = (signed int*)0x8AAA;//
canal3 = (signed int*)0x9554;//

}

leeA1 = (unsigned int*)0x07108;

*(canal1++) = ((*(leeA1++))>>4)-1757;// mic1-
mic8-----------Qi = 12 Canal DERECHO
*(canal2++) = ((*(leeA1++))>>4)-2047;// Canal
IZQUIERDO
*(canal3++) = ((*(leeA1++))>>4)-1875;// Canal
TRASERO

con1++;

if (con1 == 2731){

con2++;

con1 = 1;

if (con2 == 3){

con2 = 1;

}

}


AdcRegs.ADCTRL2.bit.RST_SEQ1=0x1;
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
return;
}
```

Apéndice B

Simulación de Arreglo

Para el diseño del sistema se utilizó de base una simulación la cual se programo en matlab, a continuación se muestra el código correspondiente

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PROGRAMA SIMULACIÓN DE ARREGLO PLANAR DE
TRES MICRÓFONOS EN UN ESPACIO%%
%%                              HOMOGENEO
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% vértices
D = 0.17; %distancia entre micrófonos

psi = 90; %ángulo de inclinación del plano en
ejes z,y

ro = 1; %DISTANCIA FUENTE AL ORIGEN

psi = psi * pi/180;

vnormal = [0 cos(psi) sin(psi)]; %vector
normal del plano

%VERTICES DEL ARREGLO
x1 = [0 D/2];
y1 = [0 (D/2)*tan(30*pi/180)*vnormal(3)];
z1 = [0 -(D/2)*tan(30*pi/180)*vnormal(2)];

x2 = [0 -D/2];
y2 = [0 (D/2)*tan(30*pi/180)*vnormal(3)];
z2 = [0 -(D/2)*tan(30*pi/180)*vnormal(2)];

x3 = [0 0];
y3 = [0 (D/2)*tan(30*pi/180)*vnormal(3)-
((3^0.5)/2)*D*vnormal(3)];
z3 = [0 ((3^0.5)/2)*D*vnormal(2)-
(D/2)*tan(30*pi/180)*vnormal(2)];

x4 = [x1(2) x2(2)]; %líneas que unen los
vértices, línea vértice 1 a 2
y4 = [y1(2) y2(2)];
z4 = [z1(2) z2(2)];

x5 = [x2(2) x3(2)]; %líneas que unen los
vértices, línea vértice 2 a 3
y5 = [y2(2) y3(2)];
z5 = [z2(2) z3(2)];

x6 = [x3(2) x1(2)]; %líneas que unen los
vértices, línea vértice 3 a 1
y6 = [y3(2) y1(2)];
z6 = [z3(2) z1(2)];

vx1 = [0 vnormal(1)]; %vector normal
vy1 = [0 vnormal(2)];
vz1 = [0 vnormal(3)];

fi   = [ 0 : 5 : 360 ]; % elevación
teta = [ 0 : 5 : 360 ]; % azimuth

%coordenadas de la fuente

x = ro * sin(fi*pi/180)' * cos(teta*pi/180);
y = ro * sin(fi*pi/180)' * sin(teta*pi/180);
z = ro * cos(fi*pi/180);

unos = ones(1,length(z));

z = z' * unos;

%cálculo de distancia a cada vértice

v1 = [2*(D/3)*cos(45*pi/180) -
(D/3)*cos(45*pi/180) -(D/3)*cos(45*pi/180)];

v2 = [-(D/3)*cos(45*pi/180)
2*(D/3)*cos(45*pi/180) -
(D/3)*cos(45*pi/180)];

v3 = [-(D/3)*cos(45*pi/180) -
(D/3)*cos(45*pi/180) 2*(D/3)*cos(45*pi/180)];

%pueden servir para campo cercano
```

```
d1 = ( (x - x1(2)).^2 + (y - y1(2)).^2 + (z -
z1(2)).^2 ).^0.5;

d2 = ( (x - x2(2)).^2 + (y - y2(2)).^2 + (z -
z2(2)).^2 ).^0.5;

d3 = ( (x - x3(2)).^2 + (y - y3(2)).^2 + (z -
z3(2)).^2 ).^0.5;

%para campo lejano se consideran planos
%ecuación paramétrica recta entre vértice uno
y frente de onda plano
t1 = 1 - (( x*x1(2) + y*y1(2) + z*z1(2) ) ./
(x.^2 + y.^2 + z.^2 ) );
xd1 = x1(2) + t1 .*x ;
yd1 = y1(2) + t1 .*y ;
zd1 = z1(2) + t1 .*z ;

W = 1; %potencia de la señal

c = 340; %velocidad del sonido m/s

%distancia vértice a frente de onda plano
d11 = ( ((x1(2) - xd1).^2) + ((y1(2) -
yd1).^2) + ((z1(2) - zd1).^2)).^0.5;

T1 = d11/c; % tiempo entre frente de onda
plano al vértice 1

%ecuación paramétrica recta entre vértice dos
y frente de onda plano
t2 = 1 - (( x*x2(2) + y*y2(2) + z*z2(2) ) ./
(x.^2 + y.^2 + z.^2 ) );
xd2 = x2(2) + t2 .*x ;
yd2 = y2(2) + t2 .*y ;
zd2 = z2(2) + t2 .*z ;

d22 = ( ((x2(2) - xd2).^2) + ((y2(2) -
yd2).^2) + ((z2(2) - zd2).^2)).^0.5;

T2 = d22/c; % tiempo entre frente de onda
plano al vértice 2

%ecuación paramétrica recta entre vértice
tres y frente de onda plano
t3 = 1 - (( x*x3(2) + y*y3(2) + z*z3(2) ) ./
(x.^2 + y.^2 + z.^2 ) );
xd3 = x3(2) + t3 .*x ;
yd3 = y3(2) + t3 .*y ;
zd3 = z3(2) + t3 .*z ;

d33 = ( ((x3(2) - xd3).^2) + ((y3(2) -
yd3).^2) + ((z3(2) - zd3).^2)).^0.5;
%atenuación debida a la distancia
T3 = d33/c; % tiempo entre frente de onda
plano al vértice 3

a1 = W./(4*pi*(d11.^2)); %atenuación entre
frente de onda plano al vértice 1
a2 = W./(4*pi*(d22.^2)); %atenuación entre
frente de onda plano al vértice 2
a3 = W./(4*pi*(d33.^2)); %atenuación entre
frente de onda plano al vértice 3

Fs = 48000; %frecuencia de muestreo

f = 400; %frecuencia del tono

n = [1:160]; %número de muestras

TOL = 50; %muestras que se descartan al
principio de grabación

LONGT = 47000; %longitud del vector a usar

m = 19; %dirección de elevación de la fuente

p = 28; %azimuth de elevación de la fuente

menor = min([d11(m,p) d22(m,p) d33(m,p)]);

R = a1(m,p)*sin((2*pi*f)*((1/Fs)*n -
((d11(m,p) - menor)/c)));

L = a2(m,p)*sin((2*pi*f)*((1/Fs)*n -
((d22(m,p) - menor)/c)));

B = a3(m,p)*sin((2*pi*f)*((1/Fs)*n -
((d33(m,p) - menor)/c)));

R1 = a1(m,p)*calculoSinc(wx,((d11(m,p) -
menor)/c)*Fs);

L1 = a2(m,p)*calculoSinc(wx,((d22(m,p) -
menor)/c)*Fs);

B1 = a3(m,p)*calculoSinc(wx,((d33(m,p) -
menor)/c)*Fs);

% ventanas

tam = 128;
w1 = window(@hamming, 128);
w2 = window(@hamming, 256);
numVen = 106;

tamVec = length(R1);

N = floor(tamVec/tam);

if numVen > N

    numVen = N;

end

L1R1 = xcorr(L1((((numVen-
1)*tam)+1):(tam*numVen)).*(w1'),R1((((numVen-
1)*tam)+1):(tam*numVen)).*(w1'));

R1B1 = xcorr(R1((((numVen-
1)*tam)+1):(tam*numVen)).*(w1'),B1((((numVen-
1)*tam)+1):(tam*numVen)).*(w1'));

B1L1 = xcorr(B1((((numVen-
1)*tam)+1):(tam*numVen)).*(w1'),L1((((numVen-
1)*tam)+1):(tam*numVen)).*(w1'));

angulos = [90 73.4022 66.4435 61.0450 56.4427
52.3415 48.5904 45.0995 41.8103 38.6822
35.6853 32.7972 30 27.2796 24.6243 22.0245
19.4712 16.9578 14.4775 12.0247 9.5941 7.1808
4.7802 2.388 0];

L1R1 = L1R1(tam-24:tam+24);

R1B1 = R1B1(tam-24:tam+24);

B1L1 = B1L1(tam-24:tam+24);
```

```
angL1R1 = [-180+angulos(end:-1:2) -angulos
angulos(end-1:-1:1) 180-angulos(2:end-1)];

angR1B1 = 120+[-180+angulos(end:-1:2) -
angulos angulos(end-1:-1:1) 180-
angulos(2:end-1)];

angB1L1 = -120+[-180+angulos(end:-1:2) -
angulos angulos(end-1:-1:1) 180-
angulos(2:end-1)];

aL1R1 = [ L1R1(25:-1:2) L1R1 L1R1(end-1:-
1:26) ];

aR1B1 = [ R1B1(25:-1:2) R1B1 R1B1(end-1:-
1:26) ];

aB1L1 = [ B1L1(25:-1:2) B1L1 B1L1(end-1:-
1:26) ];

%plot(angL1R1, aL1R1)

%hold all

angR1B1 = [ (angR1B1(70:end)-360)
angR1B1(1:69)];

aR1B1 = [aR1B1(70:end) aR1B1(1:69)];

%plot(angR1B1, aR1B1)

angB1L1 = [angB1L1(29:end)
(angB1L1(1:28)+360)];

aB1L1 = [aB1L1(29:end) aB1L1(1:28)];

%plot(angB1L1, aB1L1)

%figure(4); plot(angL1R1,aR1B1+aB1L1+aL1R1) %
localizacion por cross correlación

tam = 256;

numVen = 106;

tamVec = length(R1);

N = floor(tamVec/tam);

if numVen > N

    numVen = N;

end

r = R1((((numVen-
1)*tam)+1):(tam*numVen)).*(w2');

l = L1((((numVen-
1)*tam)+1):(tam*numVen)).*(w2');

b = B1((((numVen-
1)*tam)+1):(tam*numVen)).*(w2');

R = fft(r);

L = fft(l);

B = fft(b);

LR =
(L.*(conj(R))./(((abs(R))).*((abs(L))));

RB = (R.*(conj(B))./((abs(R)).*(abs(B)));

BL = (B.*(conj(L))./((abs(B)).*(abs(L)));

LR = LR.^2;

RB = RB.^2;

BL = BL.^2;

lrp = fft(real(LR.^2) - i*imag(LR.^2));

rbp = fft(real(RB.^2) - i*imag(RB.^2));

blp = fft(real(BL.^2) - i*imag(BL.^2));

lrp = real(lrp.^2) - i*imag(lrp.^2);

rbp = real(rbp.^2) - i*imag(rbp.^2);

blp = real(blp.^2) - i*imag(blp.^2);

lr = ifft(LR);

rb = ifft(RB);

bl = ifft(BL);

lr = [lr(end-23:end) lr(1:25)];

rb = [rb(end-23:end) rb(1:25)];

bl = [bl(end-23:end) bl(1:25)];

if lr(25) < 0.95     % umbral para descartar
pico al principio de las muestras

    lr(25) = 0;

end

if rb(25) < 0.95

    rb(25) = 0;

end

if bl(25) < 0.95

    bl(25) = 0;

end

alr = [lr(25:-1:2) lr lr(end-1:-1:26)];

arb = [rb(25:-1:2) rb rb(end-1:-1:26)];

abl = [bl(25:-1:2) bl bl(end-1:-1:26)];

arb = [arb(70:end) arb(1:69)];

abl = [abl(29:end) abl(1:28)];

plot(angL1R1,alr)      % GRÁFICAS DE
LOCALIZACIÓN POR PHAT

hold all
```

```
plot(angL1R1,arb)

plot(angL1R1,abl)

figure(2); plot(angL1R1,alr+arb+abl)

%plot(bl)

%figure(1); plot3(x1, y1, z1)% trazo de
arreglo planar triangular

%hold all

%figure(1); plot3( x2, y2, z2)

%figure(1); plot3( x3, y3, z3)

%figure(1); plot3( x4, y4, z4)

%figure(1); plot3( x5, y5, z5)

%figure(1); plot3( x6, y6, z6)

%figure(1); plot3( vx1, vy1, vz1) % trazo de
la normal al plano

%figure(1); plot3( x, y, z) % trazo esfera de
puntos

%cont = 1;

% GRAFICA DE DIRECCIONES DE FRENTE DE ONDA

%while cont <= 73

%figure(1); plot3([0 x(m,p)],[0 y(m,p)], [0
z(m,p)]) % direccion de frente de onda

%cont = cont + 1;

%end

%plot(angulos, estB1R1)
```

Manipulación de Datos

```
function v = manArchivos(sName)%%abre archivo
y devuelve 21 archivos para los tres canales

FID = fopen(sName);

if length(sName) > 4
sName = sName(1:end-4);
end

A = fread(FID,inf);

cont = 1;

tam = length(A); %%tamaño de vector

while cont <= 21

    if cont <= 9

        eval(['FID' num2str(cont) '= fopen('
char(39) sName '0' num2str(cont) '.dat'
char(39) ', ' char(39) 'w' char(39) ');']);

    else

        eval(['FID' num2str(cont) '= fopen('
char(39) sName num2str(cont) '.dat' char(39)
', ' char(39) 'w' char(39) ');']);

    end

   eval(['fwrite(FID' num2str(cont)
',A(1:20));']);

    cont = cont + 1;

end

cont = 21;

cont2 = 1;

cont3 = 1;

cont4 = 1;

par = 0;

B = 0;

while cont <= tam

    if ( A(cont) == 13 )&&( A(cont+1) == 10 )

        par = par + 1;

        cont2 = cont2 + 1;

        cont4 = cont4 + 1;

    end

    if par == 2

        BX = [A(cont-6:cont-1); A(cont-
12:cont-9); A(cont:cont+1)];

        if cont2 <= 3

            BW = BX;

        else

            BW = [BW; BX];

        end

        par = 0;

    end

    if cont2 == 129

        eval(['fwrite( FID' num2str(cont3) ',
BW);']);

        cont2 = 1;

        cont3 = cont3 + 1;

    end
```

```
    if cont4 == 2732

       cont2 = 1;

       cont3 = 1;

       cont4 = 1;

    end

    cont = cont + 1;

end

fclose('all');

v = 0;

end
```

Manipulación de Datos 2

```
function v = manArchivos2(sName)%%abre
archivo y devuelve 21 archivos para los tres
canales

FID = fopen(sName);

if length(sName) > 4
sName = sName(1:end-4);
end

A = fread(FID,inf);

cont = 1;

tam = length(A); %%tamaño de vector

while cont <= 10

    if cont <= 9

        eval(['FID' num2str(cont) '= fopen('
char(39) sName '0' num2str(cont) '.dat'
char(39) ', ' char(39) 'w' char(39) ');']);

    else

        eval(['FID' num2str(cont) '= fopen('
char(39) sName num2str(cont) '.dat' char(39)
', ' char(39) 'w' char(39) ');']);

    end

   eval(['fwrite(FID' num2str(cont)
',A(1:20));']);

    cont = cont + 1;

end

cont = 21;

cont2 = 1;

cont3 = 1;

cont4 = 1;

B = 0;

while cont <= tam

    if ( A(cont) == 13 )&&( A(cont+1) == 10 )

        cont2 = cont2 + 1;

        cont4 = cont4 + 1;

        BX = [A(cont-6:cont-5); [char(48);
char(48); char(48); char(48)]; A(cont-4:cont-
1); A(cont:cont+1)];

        if cont2 <= 2

            BW = BX;

        else

            BW = [BW; BX];

        end

    end

    if cont2 == 257

        eval(['fwrite( FID' num2str(cont3) ',
BW);']);

        cont2 = 1;

        cont3 = cont3 + 1;

    end

    if cont4 == 2732

       cont2 = 1;

       cont3 = 1;

       cont4 = 1;

    end

    cont = cont + 1;

end

fclose('all');

v = 0;

end
```