



**DOCUMENTO TÉCNICO DE DESARROLLO DE ALGORITMOS DE MUSIC INFORMATION RETRIEVAL (EXTRACCIÓN DE INFORMACIÓN MUSICAL) EN EL MARCO DEL PROYECTO: OPTIMIZACIÓN DE PROCESOS DE PRODUCCIÓN DE MÚSICA POP, BASADO EN MODELOS COMPUTACIONALES**

CARLOS ALBERTO RODRIGUEZ

RUBEN CORRALES

JORGE CORRALES

CARLOS DIEGO FERRIN BOLAÑOS

INSTITUCIÓN UNIVERSITARIA ANTONIO JOSÉ CAMACHO

FACULTADA DE INGENIERÍA

SANTIAGO DE CALI

2016

INTRODUCCIÓN

*LEYENDA: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit.*

Contenido

[Contenido 2](#_Toc469241047)

[Listado de Figuras 4](#_Toc469241048)

[Listado de Tablas 5](#_Toc469241049)

[1. OBJETIVOS 6](#_Toc469241050)

[2. MARCO DE REFERENCIA 6](#_Toc469241051)

[2.1 MIR y Procesado de señales 6](#_Toc469241052)

[2.1.1 Señales de audio 7](#_Toc469241053)

[2.1.2 Procesado digital de señales 12](#_Toc469241054)

[2.1.3 Análisis en el dominio del tiempo 12](#_Toc469241055)

[2.1.4 Análisis en el dominio de la frecuencia 22](#_Toc469241056)

[2.1.5 Análisis en tiempo-frecuencia 26](#_Toc469241057)

[2.2 Análisis Tonal 28](#_Toc469241058)

[2.2.1 Percepción humana del “Pitch” 28](#_Toc469241059)

[2.2.2 Representación del “Pitch” en Música 28](#_Toc469241060)

[2.2.3 Detección de frecuencia fundamental 28](#_Toc469241061)

[2.2.4 Estimación de frecuencia de sintonización 29](#_Toc469241062)

[2.2.5 Detección de notas 29](#_Toc469241063)

[2.2.6 Reconocimiento de acordes 29](#_Toc469241064)

[2.3 Análisis Temporal 29](#_Toc469241065)

[2.3.1 Percepción humana de los eventos temporales 29](#_Toc469241066)

[2.3.2 Representación de eventos temporales en música 29](#_Toc469241067)

[2.3.3 Detección de “Onset” 29](#_Toc469241068)

[2.3.4 Histograma de golpes (beat) 29](#_Toc469241069)

[3. TÉCNICAS DE MUSIC INFORMATION RETRIEVAL 29](#_Toc469241070)

[3.1 Bibliotecas MIR 30](#_Toc469241071)

[3.1.1 Requerimientos Hardware 30](#_Toc469241072)

[3.1.2 Requerimientos Software 30](#_Toc469241073)

[3.2 Funcionamiento de la GUI para MIR 30](#_Toc469241074)

[3.2.1 Módulo de detección de notas 30](#_Toc469241075)

[3.2.2 Módulo de reconocimiento de acordes 30](#_Toc469241076)

[4. CONCLUSIÓN 30](#_Toc469241077)

Listado de Figuras

[Figura 1 . Pipeline para Music Information Retrieval, MIR. 7](#_Toc469241078)

[Figura 2. Señal Analógica, Y(t)*,* vs Señal Digital, y[n]. 8](#_Toc469241079)

[Figura 3. Comandos ejecutados en Matlab para cargar una señal de audio a partir de un archivo de audio. Ventanas de Variables (1), Workspace (2) y Comandos (3). 10](#_Toc469241080)

[Figura 4. Amplitud de la señal de audio vs tiempo. 11](#_Toc469241081)

[Figura 5. Salida de ejecución del Script con bucle iterativo u operadores sobrecargados para operación de escalonado. 14](#_Toc469241082)

[Figura 6. Salida de ejecución del script de submuestreo. 16](#_Toc469241083)

[Figura 7. Aplicación de un filtro de media móvil con L = 100. 19](#_Toc469241084)

[Figura 8. Derivada de una señal. 20](#_Toc469241085)

[Figura 9. Integración de una señal. 22](#_Toc469241086)

[Figura 10. Obtención de la DFT de una señal, gráfica del plano complejo, parte real e imaginaria. 25](#_Toc469241087)

[Figura 11. Obtención de la DFT de una señal, gráfica en el plano complejo, su magnitud y fase en función de la frecuencia. 26](#_Toc469241088)

[Figura 12. Algoritmo de la STFT. 28](#_Toc469241089)

Listado de Tablas

[Tabla 1. Cáluculos para obtención la convolución entre h e y. 17](#_Toc469241090)

# OBJETIVOS

● Recopilar información bibliográfica sobre el desarrollo de técnicas de Music Information Retrieval: Artículo, Libros, Proyectos Open Source, Dataset.

● Identificar metodologías y técnicas de Music Information Retrieval para el análisis de audio.

● Diseñar e implementar algoritmos de Music Information Retrieval para el análisis de audio.

# MARCO DE REFERENCIA

*Este capítulo introduce y resume las más importantes características de los sistemas de procesamiento de señales de audio y de los sistemas MIR.*

## MIR y Procesado de señales

La Recuperación de Información Musical o MIR (por sus siglas en inglés, *Music Information Retrieval*) es un campo interdisciplinar que busca la recuperación o extracción de información a partir de señales de audio musical. Es considerado un campo en constante crecimiento con muchos casos de éxito en aplicaciones del mundo real, tales como las aplicaciones móviles ***Shazam***, ***Spotify***, etc.; que permiten la identificación automática del título de una canción escuchada a través del micrófono del celular o la búsqueda y organización inmediata de canciones en grandes bases de datos. MIR es un campo que involucra disciplinas como la Musicología, Piscología, Procesado de Señales, Aprendizaje de Máquina (*Machine Learning*).

El pipeline general de los algoritmos MIR puede verse en la Figura 1. La adquisición de la señal de audio y la información musical constituyen los parámetros de entrada y salida respectivamente de los sistemas MIR. Los bloques de segmentación, extracción de características y *machine learning* son las etapas principales dentro del proceso de desarrollo de sistemas MIR. La segmentación divide la señal en sub-señales. El tamaño, el tipo de enventanado y el solape son parámetros que determinarán los resultados siguientes. El bloque de extracción de características obtiene y reduce la dimensionalidad de patrones de bajo, medio y alto nivel. Finalmente, la etapa de *machine learning* permite modelar, junto con los patrones obtenidos anteriormente, la forma como el ser humano percibe la música y realiza inferencias sobre el audio musical obteniendo información relevante.

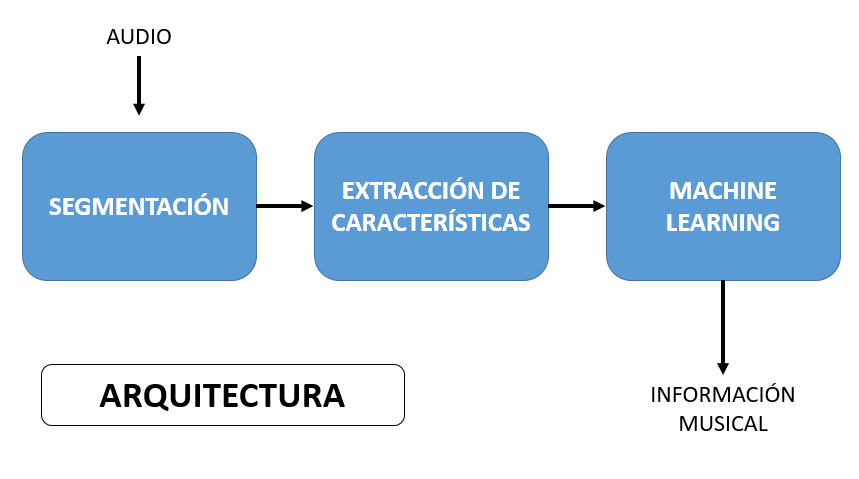


Figura 1 . Pipeline para Music Information Retrieval, MIR.

### Señales de audio

Una señal de audio, tal como la perciben los humanos puede ser descrita como una función del nivel de presión del sonido dependiente del tiempo. Para capturar esta señal es posible utilizar un el cual convierte los niveles de presión del sonido a voltajes. Esta señal de voltaje está definida para todos los instantes de tiempo y por lo tanto es considerado un señal de tiempo continuo Y(t). Para poder manipular Y(t) en un computador es necesario capturar una versión de la misma representable en el mundo digital. Esto es Y(t)se debe convertir en y[n], donde esta última es una versión de **tiempo discreto** y de **valores cuantizados** de Y(t).

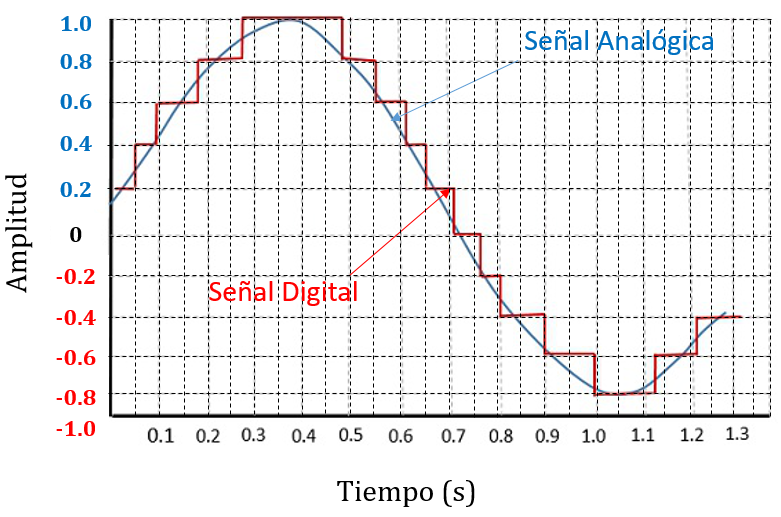


Figura 2. Señal Analógica, Y(t)*,* vs Señal Digital, y[n].

Una señal discreta en el tiempo significa que la señal solo está definida para ciertos instantes de tiempo. Esto es, si la señal fue muestreada (Frecuencia de Muestreo, ***Fs***) a 20 Hz, es decir veinte muestras en un segundo, cada una de las muestras está separada de la otra en 0.05 s. En la Figura 2 se puede observar el ejemplo de una señal analógica muestreada a ***Fs*** = 20 Hz.

Este tiempo se conoce como periodo de muestreo, ***Ts***. La relación entre la frecuencia de muestreo y el periodo de muestreo viene dada por la siguiente ecuación ():

|  |  |  |
| --- | --- | --- |
|  |  | () |

El ser humano solo puede percibir componentes frecuenciales en la señal desde los 20 a los 20.000 Hz. Luego entonces es necesario muestrear este tipo de señales por lo menos al doble de la máxima frecuencia (**Teorema de Nyquist**), esto es, a 40.000 Hz. En la práctica, es muy común encontrar que las señales de audio musical se muestrean a 44.100 Hz. Lo anterior significa que la separación temporal entre muestras es de 2.2676x10-5 s (0.02267 ms ≈ 22.7 µs).

Por otro lado, la cuantización hace referencia a que los valores de amplitud de la señal de audio también se han discretizados al igual que el tiempo. Usualmente, la amplitud se fija dentro de un rango, entre un valor máximo, Amax, y uno mínimo, Amin, simétrico respecto al valor cero. Este rango se divide en M partes. Por ejemplo, en la Figura 2 se ha dividido el rango de -1.0 a 1.0 en 11 partes simétricas respecto al 0. Si M es una potencia de 2, la amplitud de cada muestra cuantizada puede ser fácilmente representada mediante un código binario de longitud, NB, igual a log2(M) bits. Dado que en este escenario M sería siempre un número impar, es imposible conservar al cero como un valor de cuantización y a la vez tener un número simétrico de pasos de cuantización. En la práctica se fija un paso menos en los valores positivos. Longitudes típicas de códigos binarios en señales de audio son 16, 24 y 32 bits. Estas longitudes permiten tener 216 = 65536, 224 = 16.777.216 y 232 = 4.294.967.297 divisiones del rango, respectivamente. A su vez se tienen, por ejemplo, separaciones entre valores de amplitud cuantizadas, ΔQ, para un rango entre -1.0 y 1.0, y NB = 32 de:

|  |  |  |
| --- | --- | --- |
|  |  | () |

Ejemplo:

Generalmente, las señales de audio pueden ser mono (un canal) o estéreo (dos canales). Para propósitos de desarrollo de algoritmos MIR se utilizarán, en este documento, señales de audio de un canal. En caso de disponer de señales audio estéreo se seleccionará uno de los canales de esta señal.

El comando **audioread** de Matlab permite capturar información de los datos de la señal de audio, la frecuencia de muestreo desde un archivo de audio (MP3, WAV, M4A, MP4, OGG, FLAC).

> > [signal, Fs] = **audioread**(filename); *reads an audio file specified by the string “filename”, returning the sampled data in “signal” and the sample rate “Fs”, in Hertz.*

Si “signal” es de dos canales, se puede seleccionar el canal 1 o el canal 2 mediante la siguiente instrucción en Matlab.

> > y = signal(:, 1); **% Canal 1**

> > y = signal(:, 2); **% Canal 2**

Los datos en la variable “y” pueden observarse haciendo doble clic en la variable en el Workspace de Matlab, ver Figura 3.

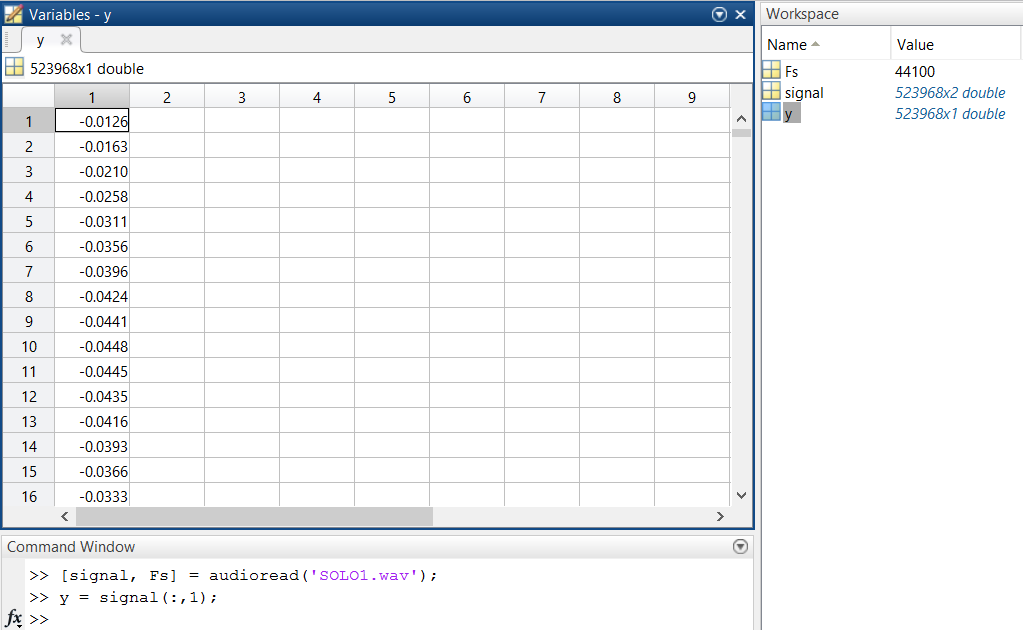


Figura 3. Comandos ejecutados en Matlab para cargar una señal de audio a partir de un archivo de audio. Ventanas de Variables (1), Workspace (2) y Comandos (3).

En el Workspace de Matlab, Figura 3, puede observarse también que el número de datos presentes en el archivo de audio son 523.968. Si la separación entre muestras es de aproximadamente de 22.7 µs (Ts = 1/Fs = 1/44100), entonces se puede crear una escala de tiempo desde 0 segundos hasta (523.967 x Ts segundos). En Matlab esta escala de tiempo puede crearse mediante la siguiente instrucción:

> > [signal, Fs] = **audioread**(‘SOLO1.wav’); **% Cargando el archivo de audio.**

> > y = signal(:, 1); **% Obteniendo el Canal 1.**

> > NumDatos = **length**(y); **% Obteniendo el número de datos.**

> > time = 0 **:** 1/Fs **:** (NumDatos - 1)/Fs; **% Vector de Instantes de Tiempo.**

Para visualizar los datos mediante un gráfico se puede utilizar el comando **plot**. Se recomienda ejecutar > > help **plot** en la ventana de comandos en Matlab para ver todas las opciones que ofrece esta función.

> > **plot**(time, y); **% Graficando time vs signal.**

> > **xlabel**(‘Tiempo (s)’); **% Etiqueta para el Eje X.**

> > **ylabel**(‘Amplitud’); **% Etiqueta para el Eje Y.**

> > **grid on**; **% Activando la grilla sobre los ejes X e Y.**

Las anteriores instrucciones generará el gráfico la siguiente figura:



Figura 4. Amplitud de la señal de audio vs tiempo.

### Procesado digital de señales

El procesado de señales es la manipulación matemática de una señal digital sobre un sistema digital de cómputo. Esta manipulación matemática de la señal puede realizarse en el dominio del tiempo, en el dominio de la frecuencia o en el dominio tiempo-frecuencia de la señal.

### Análisis en el dominio del tiempo

Existe una gran cantidad de literatura alrededor del procesado digital de señales, no se pretende por ningún motivo en este documento hacer una descripción completa de esta temática, más bien, se proveerá un breve resumen al respecto enfatizando en implementaciones en Matlab.

Todos los audios procesados así como el original pueden ser reproducidos mediante las siguientes instrucciones en Matlab:

> > p = **audioplayer**(y, Fs); **% Creando objeto de reproducción.**

> > **play**(p); **% Reproduciendo.**

**Escalamiento en Amplitud:** Esta operación permite amplificar o reducir en amplitud toda la señal o porciones de la misma. Es muy utilizada cuando la reproducción del audio presenta un volumen bajo, y se requiere aumentar el volumen para poder escucharlo a través de un dispositivo de salida como un parlante. En la siguiente ecuación se puede observar que la amplificación o reducción depende de si el parámetro α en valor absoluto es mayor o menor que 1. Valores negativos de α realizan una reflexión en torno al eje del tiempo a la señal.

|  |  |  |
| --- | --- | --- |
|  |  | () |

A continuación se muestran dos script en Matlab para reducir la amplitud de la señal a un 50 %. El primero de ellos utiliza un bucle iterativo y el segundo aprovecha el poder de los operadores sobrecargados en Matlab. En la Figura 5 se puede observar la salida de ejecución de cualquiera de los scripts.

**Script para escalonamiento 1: Bucle Iterativo**

[signal, Fs] = audioread('SOLO1.wav');

y = signal(:, 1);

time = 0:1/Fs:(length(signal) - 1)/Fs;

NumDatos = length(y);

yo = zeros(1, NumDatos);

**% Bucle iterativo**

for i=1:NumDatos

yo(i) = 0.5\*y(i);

end

**%%%%%%%%%%%%%%%%%%%%%%%%%%**

figure(1);

subplot(1,2,1); plot(time, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original');

subplot(1,2,2); plot(time, yo);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Escalonada');

**Script para escalonamiento 2: Operadores sobrecargados**

[signal, Fs] = audioread('SOLO1.wav');

y = signal(:, 1);

time = 0:1/Fs:(length(signal) - 1)/Fs;

**% Operadores sobrecargados**

yo = 0.5\*y(i);

**%%%%%%%%%%%%%%%%%%%%%%%%%%**

figure(1);

subplot(1,2,1); plot(time, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original.');

subplot(1,2,2); plot(time, yo);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Escalonada en Amplitud.');



Figura 5. Salida de ejecución del Script con bucle iterativo u operadores sobrecargados para operación de escalonado.

**Desplazamiento:** Los desplazamientos permiten correr la señal en el eje del tiempo. Se puede atrasar o adelantar la señal dependiendo del valor c. Si c es positivo la señal se adelante y si es negativo entonces se retrasa.

|  |  |  |
| --- | --- | --- |
|  |  | () |

**Reflexión:** Esta operación tiene es muy útil para implementar otras operaciones de mayor capacidad de análisis como la convolución. Mediante esta operación se obtiene una reflexión de la señal en torno a un eje paralelo al de la amplitud, pero ubicado en c.

|  |  |  |
| --- | --- | --- |
|  |  | () |

**Escalamiento en tiempo:** La aplicación principal de esta operación, ecuación (6), es la de realizar submuestreo sobre la señal. Esto significa que una señal muestreada a 44.100 Hz puede submuestrearse para obtener una señal equivalente muestreada a 22.050 Hz. Esto significa entonces que la máxima frecuencia que se podría resolver a través de un análisis espectral sería de 11.025 Hz. β debe ser un entero positivo mayor que 1.

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

A través del siguiente script se puede implementar un submuestreo a 22.050 Hz de una señal de audio muestreada a 44.100 Hz.

En la Figura 6 se observa la señal original y la submuestreada 1000 veces menos que la frecuencia original.

**Script para Submuestreo 1:**

clear all; clc; close all;

[signal, Fs] = audioread('SOLO1.wav');

FactorSubmuestreo = 1000;

y = signal(:,1);

Fs2 = Fs/FactorSubmuestreo;

time1 = 0:1/Fs:(length(signal) - 1)/Fs;

time2 = … 0:1/Fs2:(round(length(signal)/FactorSubmuestreo) - 1)/Fs2;

NumDatos = length(y);

yo = zeros(1, round(NumDatos/FactorSubmuestreo));

NumDatos2 = length(yo);

for i=1:NumDatos2

n = FactorSubmuestreo\*i;

if n>NumDatos;

break;

end

yo(i) = y(FactorSubmuestreo\*i);

end

figure(1);

subplot(1,2,1); plot(time1, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original');

subplot(1,2,2); plot(time2, yo);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Submuestreada');



Figura 6. Salida de ejecución del script de submuestreo.

**Convolución:**

Es una operación matemática entre dos señales, en este caso la señal de audio y una señal cuyos valores corresponden a los coeficientes, por ejemplo, de un filtro. En la ecuación (7) se puede mostrar la definición de esta operación. L es la longitud de h.

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

A continuación se muestra un ejemplo de los cálculos necesarios para L = 3.

Obsérvese el patrón interesante en esta operación. Se obtendría el mismo resultado si se reflejará h y se deslizará a lo largo del eje del tiempo. En cada paso se realizarían las multiplicaciones punto a punto y se sumaría todos los productos.

Por ejemplo:

Tabla 1. Cáluculos para obtención la convolución entre h e y.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| n |  |  | **y[n]** | | | |  |  | yo[n] |
| **0** | **0** | **1** | **2** | **1** | **3** | **0** | **0** |
| 1 | 1 | 0 | 2 |  |  |  |  |  | 2x1 = 2 |
| 2 |  | 1 | 0 | 2 |  |  |  |  | 0x1 + 2x2 = 4 |
| 3 |  |  | 1 | 0 | 2 |  |  |  | 1x1 + 0x2 + 2x1 = 3 |
| 4 |  |  |  | 1 | 0 | 2 |  |  | 1x2 + 0x1 + 2x3 = 8 |
| 5 |  |  |  |  | 1 | 0 | 2 |  | 1x1 + 0x3 = 1 |
| 6 |  |  |  |  |  | 1 | 0 | 2 | 1x3 = 3 |

Como se nota, se han añadido ceros donde n no existe. En Matlab el comando para realizar la convolución se llama **conv**.

> > y = [1, 2, 1, 3]; **% Inicializando y.**

> > h = [2, 0, 1]; **% Inicializando h.**

> > yo **= conv**(y, h) **% Realizando la convolución entre h e y.**

*yo =*

*2 4 3 8 1 3*

La función **conv** acepta un tercer parámetro que define la longitud de yo.

* Opción ‘same’, entrega un vector de la misma longitud del vector de entrada.
* Opción ‘full’, retorna un vector con todas las salidas posibles.
* Opción ‘valid’, entrega un vector con elementos solo donde hubo un solape total del filtro h con la señal.

> > y = [1, 2, 1, 3]; **% Inicializando y.**

> > h = [2, 0, 1]; **% Inicializando h.**

> > yo **= conv**(y, h, ‘same’) **% Realizando la convolución entre h e y.**

*yo =*

*4 3 8 1*

> > yo **= conv**(y, h, ‘full’) **% Realizando la convolución entre h e y.**

*yo =*

*2 4 3 8 1 3*

> > yo **= conv**(y, h, ‘valid’) **% Realizando la convolución entre h e y.**

*yo =*

*3 8*

Un filtro de media móvil es un filtro lineal cuyos elementos tienen el mismo valor e igual 1/L, donde L es la longitud del filtro. Por ejemplo un filtro de 10 elementos sería, h = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1,]. En el siguiente script se aplica in filtro de media móvil con L = 100.

**Script para Filtro de media móvil 1:**

clear all; clc; close all;

[signal, Fs] = audioread('SOLO1.wav');

y = signal(:,1);

L = 100;

h = ones(1, L)/L;

time = 0:1/Fs:(length(signal) - 1)/Fs;

NumDatos = length(y);

yo = conv(y, h, 'same');

figure(1);

subplot(1,2,1); plot(time, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original');

subplot(1,2,2); plot(time, yo);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Convolución con Filtro de Media Móvil');

En la Figura 7 se observa el resultado de aplicar un filtro de media móvil a una señal de audio.



Figura 7. Aplicación de un filtro de media móvil con L = 100.

**Derivación:** La derivación de una señal permite observar los cambios que ha tenido la misma a lo largo del tiempo. Esta se define para el caso de una señal digital mediante la ecuación (8). En el siguiente script se puede observar su implementación en Matlab. En la Figura 8 se puede observar el resultado de aplicación de una derivada.

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

**Script para derivación de señal 1:**

clear all; clc; close all;

[signal, Fs] = audioread('SOLO1.wav');

y = signal(:,1);

time = 0:1/Fs:(length(signal) - 1)/Fs;

NumDatos = length(y);

yo = zeros(1, NumDatos);

for i=2:NumDatos

yo(i) = y(i) - y(i-1);

end

figure(1);

subplot(1,2,1); plot(time, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original');

subplot(1,2,2); plot(time, yo);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Derivada');



Figura 8. Derivada de una señal.

**Integración:** La integración permite determinar el área bajo la curva, determinado número de muestras, ecuación (9). La implementación de esta ecuación puede ser de forma recursiva (**Script para integración de una señal 1**) o de forma directa (**Script para integración de una señal 2**). La Figura 9 muestra el resultado para ambos casos.

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

**Script para integración de una señal 1:**

clear all; clc; close all;

[signal, Fs] = audioread('SOLO1.wav');

y = signal(:, 1);

time = 0:1/Fs:(length(signal) - 1)/Fs;

NumDatos = length(y);

yo = zeros(1, NumDatos);

% Método recursivo

yo(1) = y(1);

for i=2:NumDatos

yo(i) = yo(i-1) + y(i);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1);

subplot(1,2,1); plot(time, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original');

subplot(1,2,2); plot(time, yo);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Integral');

**Script para integración de una señal 2:**

clear all; clc; close all;

[signal, Fs] = audioread('SOLO1.wav');

y = signal(:, 1);

time = 0:1/Fs:(length(signal) - 1)/Fs;

NumDatos = length(y);

yo = zeros(1, NumDatos);

% Método directo

for i=1:NumDatos

sum = 0;

for j = 1:i

sum = sum + y(j);

end

yo(i) = sum;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1);

subplot(1,2,1); plot(time, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original');

subplot(1,2,2); plot(time, yo);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Integral');



Figura 9. Integración de una señal.

### Análisis en el dominio de la frecuencia

Las transformaciones sobre las señales permiten representar la señal en otros dominios diferentes al del tiempo. En estos dominios se puede extraer información de la señal que no es fácil de obtener en el dominio del tiempo. Uno de esos dominios es conocido como el dominio de la frecuencia. La transformada de Fourier permite obtener la representación en el dominio de la frecuencia de las señales. Para el caso de señales digitales se utiliza más exactamente la transformada rápida de Fourier (FFT), que es una implementación de la Transformada Discreta de Fourier (DFT).

La DFT, Y[k] ∈ Cm, de la señal y[n] de longitud N, viene definida por:

|  |  |  |
| --- | --- | --- |
|  |  | () |

Donde:

|  |  |  |
| --- | --- | --- |
|  |  | () |

La transformada inversa viene dada por:

En Matlab el comando para obtener la **fft** es el siguiente:

> > Y =**fft**(y, NL);

* Si la longitud de “y” es menor que “NL”, se añaden ceros al final de “y” para obtener una longitud igual a NL, antes de obtener la **fft**.
* Si la longitud de “y” es mayor que “NL”, se trunca a “y” para obtener la longitud “NL”, antes de obtener la **fft**.

Si NL es una potencia de 2, la obtención de la **fft** presentará un mejor desempeño a si no lo es. Una forma de obtener NL a partir de la longitud N de “y”, como una potencia de 2 más cercana por encima a N, es mediante el siguiente comando:

> > NL = 2^(**nextpow2**(N));

En el dominio del tiempo, la escala temporal se obtiene mediante el siguiente comando:

> > tiempo = 0 **:** Ts **:** (NL - 1)\*Ts;

En el dominio de la frecuencia, la escala frecuencial se obtiene mediante el siguiente comando:

> > frecuencia = 0 **:** Fs/NL **:** (NL - 1)\*(Fs/NL);

En el siguiente script se obtiene la DFT de una señal de audio, se grafica en el plano complejo y su parte real e imaginaria en función de la frecuencia, ver Figura 10:

**Script para DFT 1: Parte real e imginaria**

clear all; clc; close all;

[signal, Fs] = audioread('SOLO1.wav');

Ts = 1/Fs;

y = signal(:, 1);

N = length(y, N);

Y = fft(y);

tiempo = 0:Ts:(N-1)\*Ts;

frecuencia = 0 : Fs/N : (N-1)\*(Fs/N);

figure(1);

subplot(2,2,1); plot(tiempo, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('Señal Original');

subplot(2,2,2); plot(Y, '\*');

title('Plano Complejo');

subplot(2,2,3); plot(frecuencia, imag(Y));

xlabel('Frecuencia (Hz)');

ylabel('Amplitud');

title('Parte Imaginaria');

subplot(2,2,4); plot(frecuencia, real(Y));

xlabel('Frecuencia (Hz)');

ylabel('Amplitud');

title('Parte Real');

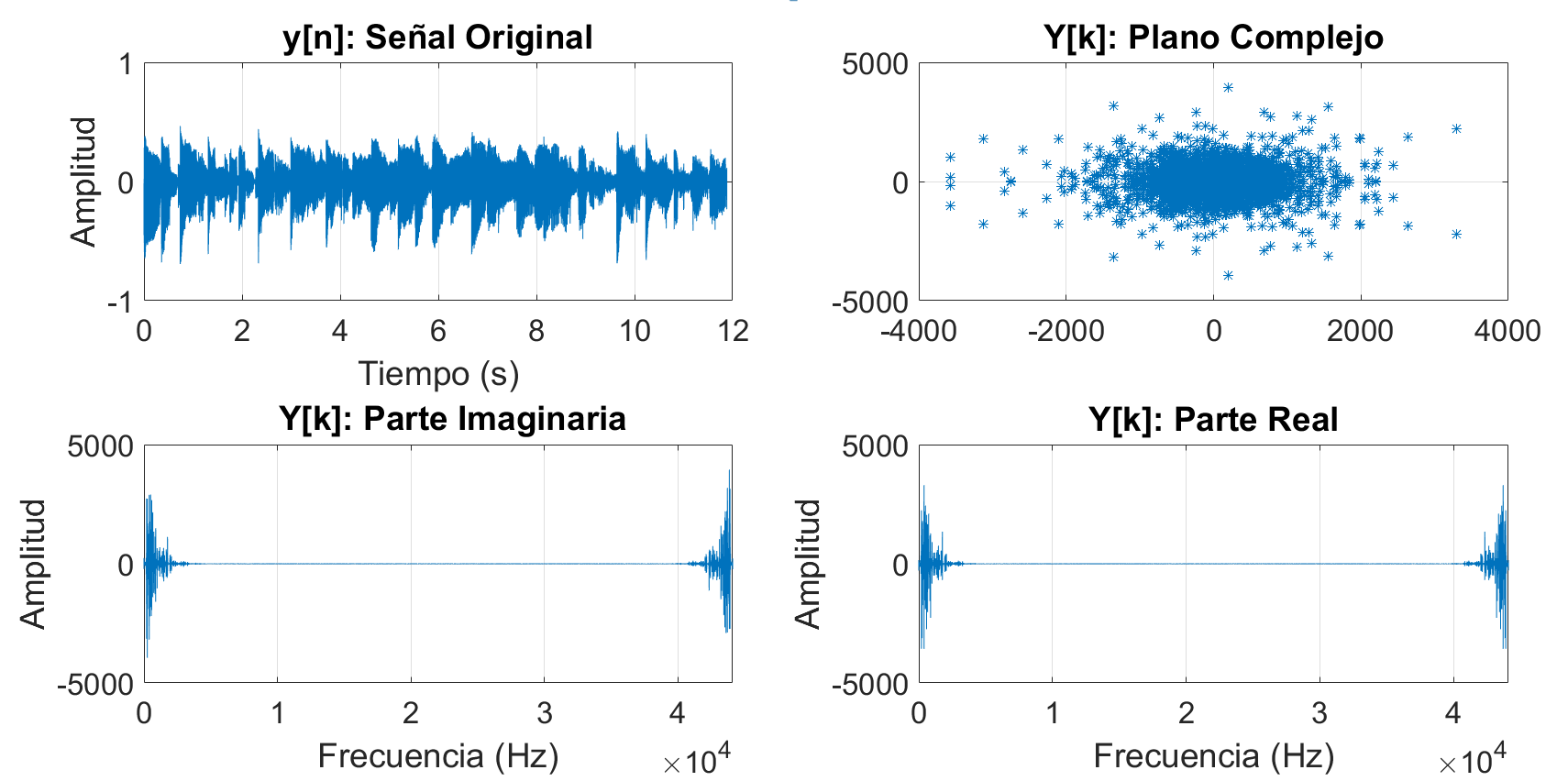


Figura 10. Obtención de la DFT de una señal, gráfica del plano complejo, parte real e imaginaria.

En el siguiente script por su parte se obtiene la magnitud y la fase, ver Figura 11.

**Script para DFT 2: Magnitud y Fase**

clear all; clc; close all;

[signal, Fs] = audioread('SOLO1.wav');

Ts = 1/Fs;

y = signal(:, 1);

Y = fft(y);

N = length(y);

tiempo = 0:Ts:(N-1)\*Ts;

frecuencia = 0:Fs/N:((N-1)\*Fs)/N;

figure(1);

subplot(2,2,1); plot(tiempo, y);

xlabel('Tiempo (s)');

ylabel('Amplitud');

title('y[n]: Señal Original');

subplot(2,2,2); plot(Y, '\*');

title('Y[k]: Plano Complejo');

subplot(2,2,3); plot(frecuencia, abs(Y));

xlabel('Frecuencia (Hz)');

ylabel('Amplitud');

title('Y[k]: Magnitud');

subplot(2,2,4); plot(frecuencia, angle(Y));

xlabel('Frecuencia (Hz)');

ylabel('Radianes');

title('Y[k]: Fase');

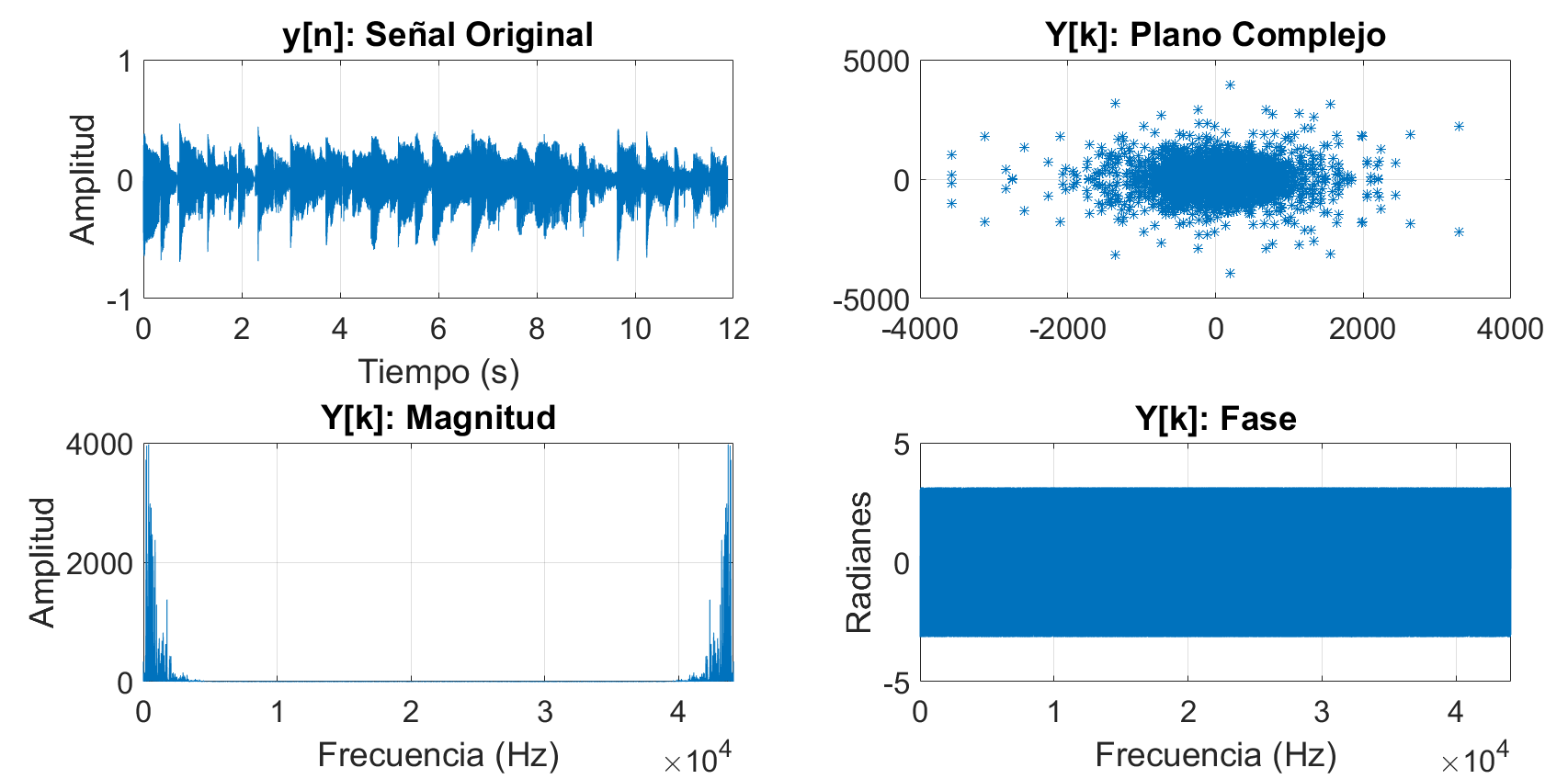


Figura 11. Obtención de la DFT de una señal, gráfica en el plano complejo, su magnitud y fase en función de la frecuencia.

La DFT de señales digitales reales es simétrica con respecto a la frecuencia de Nyquist (Fs/2), esto significa que el análisis de la señal se puede hacer hasta la frecuencia de Nyquist, sin embargo las operaciones de modificación de la DFT debe realizarse sobre todas las componentes simétricas.

La manipulación directa del espectro de señales de audio no es correcto dado que las componentes de frecuencia no son todos números enteros de frecuencia. Por otro lado, la eliminación de porciones del espectro (fijar a cero las componentes) es igual a multiplicar el espectro por una ventana rectangular, lo cual es equivalente a realizar una convolución circular de la señal con la función *sinc* (sinc[n] = n-1 sin[n]), y es sabido que esta presenta muchas oscilaciones. Por esta razón, suele utilizarse la Transformada de Fourier como herramienta para construcción de filtros, los cuales son aplicados mediante convoluciones a la señal en el dominio del tiempo. Se utilizan tanto filtros de respuesta al impulso finitas, FIR, como infinitas, IIR.

### Análisis en tiempo-frecuencia

En MIR se requiere poder analizar las señales de audio en dominios que incluyan tanto el tiempo como la frecuencia. Esto se debe a las diferentes modulaciones introducidas a lo largo del tiempo en el proceso de producción musical. La transformada de Fourier de tiempo corto, STFT, fue una de las primeras transformadas introducidas para obtener este tipo de información, tiempo-frecuencia, de las señales.

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

La STFT se puede obtener con ayuda de la FFT y el concepto de ventanas móviles. La ecuación (12) refleja esto, introduciéndose para ello el término v[n]. Por cada m se obtiene la transformada de Fourier de la señal multiplicada por una función ventana cuyo único propósito es el de evitar cambios bruscos entre transformadas contiguas. Suelen utilizarse ventanas de Hamming y Hanning.

Desde un punto de vista práctico, la STFT divide una señal es pequeños segmentos multiplicados por una función ventana, los cuales pueden ser solapados entre sí. Luego, se aplica la FFT a cada segmento de forma separada, tal como se ilustra en la Figura 12.

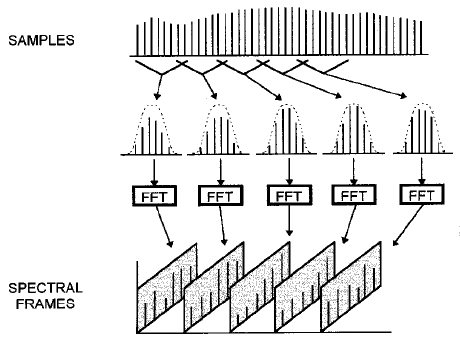


Figura 12. Algoritmo de la STFT.

## Análisis Tonal

### Percepción humana del “Pitch”

### Representación del “Pitch” en Música

### Detección de frecuencia fundamental

### Estimación de frecuencia de sintonización

### Detección de notas

### Reconocimiento de acordes

## Análisis Temporal

### Percepción humana de los eventos temporales

### Representación de eventos temporales en música

### Detección de “Onset”

### Histograma de golpes (beat)

# TÉCNICAS DE MUSIC INFORMATION RETRIEVAL

*LEYENDA: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed eiusmod tempor incidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquid ex ea commodi consequat. Quis aute iure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit.*

## Bibliotecas MIR

### Requerimientos Hardware

### Requerimientos Software

## Funcionamiento de la GUI para MIR

### Módulo de detección de notas

### Módulo de reconocimiento de acordes

# CONCLUSIÓN



REFERENCIAS