UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

ANÁLISIS E INTERPRETACIÓN DE PATRONES MUSICALES

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA :

NESTOR ABDY GARCÍA FRAGOSO

DIRECTOR DE TESIS:
Ing. Manuel Enrique Castañeda Castañeda

Ciudad Universitaria, México, 2014

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPÍTULO 1

INTRODUCCIÓN

En el mundo estamos constantemente interactuando con objetos que conforman el ambiente; este proceso llamado percepción, permite encontrar similitudes entre éstos y clasificarlos de acuerdo a sus características similares.

Este proceso no es sencillo de llevar a cabo, para ello debemos hacer uso de los sentidos para descubrir las características esenciales que definen un conjunto de objetos; primero debemos elegir un objeto cualquiera y observar su apariencia, antes de que estas características se conviertan en un patrón, debimos haberlas asociado previamente con otro objeto diferente. Finalmente el patrón definido debe percibirse recurrentemente en otros objetos diferentes pero bajo la misma regla.

En cuanto a los sonidos, el proceso es el mismo. Un cierto sonido podemos asociarlo a un objeto, una animal o una persona. Somos capaces de reconocer personas por su voz; y con la música no es diferente. Una misma nota es producida por diferentes instrumentos, sin embargo es posible identificarla en cada uno de ellos.

El objetivo de este trabajo es presentar un método computacional para poder identificar características simples de la música. La intención es presentar diferentes esquemas de reconocimiento de notas musicales, aumentando la complejidad en cada esquema. El proceso inicia con el reconocimiento de notas aisladas interpretadas con algunos instrumentos musicales, la dificultad se incrementa al agregar nuevos instrumentos y modificando la duración de cada nota de manera aleatoria.

Sin embargo, notas aisladas producidas aleatoriamente no crean música. El siguiente nivel de reconocimiento consiste en que el sistema sea capaz de adaptarse de la determinación de notas musicales aleatorias a cadenas de notas definidas, mejor conocidas como melodías.

La etapa final consiste en evaluar la precisión del sistema con un conjunto de piezas musicales de diferente ritmo, cantidad de instrumentos en canales simultáneos y velocidad.

La organización del trabajo está compuesta por diferentes capítulos. Los tres primeros capítulos comprenden conceptos generales de temas específicos relacionados con la tesis. En el capítulo cuarto se establece la obtención, organización y administración de los datos para los experimentos. El capítulo cinco describe el método principal para el reconocimiento de las notas. Capítulos seis y siete refieren el procedimiento usado por etapas, desde la preparación de los datos hasta la obtención de resultados. Finalmente están las conclusiones.

1.1 Historia de la música por etapas

La historia de la música puede dividirse en grandes periodos que se distinguen unos de otros por razones culturales, geográficas, tecnológicas y religiosas.

De acuerdo con Pelinsky (2000)

> La Etnomusicología, es una modesta disciplina o campo de trabajo en el que se investigan las significaciones que los seres humanos, en un contexto espacio-temporal determinado, asignan a la utilización del sonido que en algunas culturas se llama música (p.11).

Época	**Características**
Primitiva	• Se pueden distinguir dos tipos de música: la sagrada y la popular. • Es simple aunque suela parecer elaborada, por su falta de herramientas y métricas. • Su canto era acompasado con algún instrumento de percusión que daba mayor fuerza a la ceremonia o fiesta.
Antigua	En el antiguo Egipto, la música era de carácter religioso, y principalmente relacionada con la astronomía. El canto monódico fue la principal aportación de esta civilización. • La música griega ocupó un lugar muy importante dentro de su cultura, con un alto valor estético y ético, utilizada en la adoración de algunos dioses y en competencias deportivas. • Su principal aportación consiste en las leyes naturales que rigen el sonido, se establecieron un orden de alturas y escalas. Crearon la teoría del espacio armónico, en la que los planetas emitían un sonido en particular dependiendo su distancia a la Tierra. • En China usaban la escala pentatónica, usaban campanas de viento, e instrumentos de cuerda.
Media S. X- S. XII	• Destacan los cantos religiosos, salmos y cantos gregorianos y *la neumas*. Además nace el uso del pentagrama. • En la burguesía surgen los juglares y los trovadores, considerada música profana.
Renacimiento	• Se definen las Bella Artes, la música tiene un desarrollo lento pero continuo. • La reforma impulsó la música religiosa que otorgaba el uso de las lenguas vernáculas y la dulcificación de las estructuras musicales. • Representantes sobresalientes son Martín Lutero, y Juan Calvino.
Barroco	• Predomino la homofonía

	• Los principales aportadores en la música fueron Johann Sebastian Bach, George Frederick Haendel y Antonio Vivaldi. • Nace la opera gracias a Claudio Moteverdi con su composición Orfeo. • El violín fue uno de los instrumentos destacados.
Clasicismo	• La música cambia de residencia pasando de los palacios a los salones burgueses y finalmente a salas de conciertos públicos. • La forma de expresión preferida fue la sonata, de ella surgen el trío, la sinfonía y el concierto. • El violín y el pianoforte se convirtieron en los instrumentos fundamentales de la época. • Christoph Willibald Gluck introduce un nuevo concepto en la ópera, la tragedia, conocida como ópera bufa. • La sonata se trata de un esquema tripartito basado en una exposición. • El concierto clásico fue introducido por Mozart. Consta de tres movimientos, el primero rápido en forma de sonata, luego un aria y el último un rondó o tema. • Como compositores destacados están Franz Joseph Haydn, Wolfgang Amadeus Mozart, Ludwig Van Bethoveen.
Romanticismo	• Fernández (2002) define al romanticismo como “una reacción al siglo de la luces. El hombre romántico renuncia a ser el engranaje del mundo” (p.7). • Ya no existió la restricción en cuanto a la duración de la música, número de instrumentos y voces. • Aparecen las obras miniatura, tales como el nocturno, el improptu, el estudio y la balada. • La poseía se une a la música. • Los dramas de ópera representaban situaciones humanas. • Surge la necesidad de crear composiciones nacionales que identifiquen a los países. • Grandes compositores de la época fueron Franz Schubert, Tchaikovsky, Strauss, Chopin y Antonin Dvorak.
Impresionismo	• No sigue patrones definidos. • Cambia también drásticamente el número de instrumentos que participan en la orquesta. • Se repiten las melodías pero se cambia la intensidad del sonido
Siglo XX	• Son tantas las corrientes musicales que es imposible catalogar todos lo estilo y compositores de la época. • Destaca la música aleatoria o antirracionalismo, esta música creada a partir de la improvisación y el azar. • Con la llegada de las nuevas tecnologías nace la Música Concreta, que son grabaciones de sonidos naturales y

	reproducidos en composiciones musicales. • La Música Electrónica también tiene su origen, utilizando únicamente sonidos artificiales creados en sintetizadores acústicos. • El Jazz tiene su origen en Nueva Orleans, resultando de la fusión de la música popular negra, el ragtime y la versión blanca de la música popular afroamericana. • El Rock es otra invención de gran importancia, nace de la combinación del Blues y del Country. • Del Rock surge el Pop que se caracteriza por la utilización de instrumentos eléctricos. Destacan The Beatles, Police y Michael Jackson.

Tabla 1.1 Periodos en la Historia de la música.

1.2 Historia del cálculo computacional

En las antiguas civilizaciones del periodo clásico surgen dos detonadores que son el punto clave para el desarrollo de la computación actual: “la sistematización del razonamiento (ó como expresar razonamientos) y el desarrollo de métodos de cálculo” (Güimi, 2008, p.3).

El razonamiento sistematizado tiene su origen en la antigua Grecia durante el periodo clásico (600-300 AC), los filósofos dieron origen a las matemáticas formales. Principalmente tres filósofos que aportaron conocimiento favorable para las matemáticas formales; Platón (427-347 AC) presentó la abstracción de las ideas, Aristóteles (384-332 AC) desarrolló el razonamiento deductivo y sistematizado, y Euclides (325-265 AC) fue quien desplegó el método axiomático diferenciando entre principios (definiciones, axiomas y postulados) y teoremas.

En Babilonia surgen los métodos de cálculo, principalmente basados en métodos prueba y error, utilizados principalmente para resolver ecuaciones, originaron las tablas matemáticas como las de multiplicar. El mayor problema es el margen de error que existía en los resultados por no contar con métodos de corrección de resultados.

En el año 825, por primera vez es utilizado el término algoritmo, por los árabes. “Historiadores de la matemáticas encontraron que el origen la palabra; viene del nombre del autor de un famoso libro persa, Abu Ja’far Mohammed ibn Musa al-Khowarizmi” (Knuth, 2002, p.1).

Pero fue hasta 1202 cuando la numeración arábiga tuvo su expansión global, gracias al matemático Leonardo Fibonacci (1170-1240) con un libro (1202, Liber abaci) en el que explicaba el uso de los números y la notación posicional del cero.

Wilhelm Schickard fue el precursor de la computación digital. En 1623 diseño e inventó la primera calculadora digital, la cual podía realizar sumas y restas, sin embargo él no es considerado el padre la computación debido a que su invento no tuvo difusión por su muerte.

Con el surgimiento del cálculo 30 años más tarde, Leibniz fue quien mejoró la Pascalina añadiendo la multiplicación y la división.

En 1801, en Francia, Joseph Marie Jacquard diseña un telar, con la singularidad de que añadió tarjetas perforadas para representar patrones en las telas, así sólo debía elegir la tarjeta y la máquina se encargaba de leer la tarjeta y tejer el patrón en la tela. Estas tarjetas perforadas son consideradas como el primer mecanismo de almacenamiento de datos.

El padre de la computación fue Charles Babbage, fundador de la Royal Astronomical Society de Inglaterra. Él diseño una máquina de propósito general sin que estuviera restringida a una tarea específica. Su colega Ada Byron fue la primera en desarrollar programas para esta máquina y por lo tanto es la primera programadora de la historia.

Georges Boole (1854) tuvo una contribución destacada en su libro “Una investigación sobre las leyes de la Verdad”, en este trabajo establece:

> El proceso del razonamiento mediante una representación simbólica. Para ello utilizó variables que solo podían adoptar dos valores “1” (verdadero) y “0” (falso) …; simbolizó el operador lógico “OR” con el símbolo “±” y el símbolo “AND” con el ”*”; y realizó un estudio en profundidad del algebra de las expresiones que sólo contenían este tipo de variables.

Durante la Segunda Guerra Mundial, en Londres crearon la primera computadora diseñada para decodificar los mensajes de radio cifrados de los alemanes, este proyecto estuvo a cargo de Alan Turing y la computadora se llamaba Colossus.

En 1939 surge la ENIAC (Electronic Numerical Integrator and Calculator) proyecto organizado por Estados Unidos dirigido por John Atanasoff y Clifford Berry. Su sucesor la EDVAC (Electronic Discrete Variable Automatic Calculator) incorporó métodos matemáticos de John Von Neumann que permitía corregir algunas deficiencias de la ENIAC, y hoy en día es conocida como la arquitectura Von Neuman.

1.3 Historia de la inteligencia artificial

La primera persona en estudiar el comportamiento del cerebro desde una perspectiva computacional fue Alan Turing en el año de 1936. Pero fueron el neurobiólogo, Warren McCulloh y el matemático, Walter Pitts, quienes profundizaron sobre el comportamiento de las neuronas y su forma de trabajar simulando sus resultados en una red construida con circuitos electrónicos.

Tres años más tarde en 1949, Donald Hebb dio las bases de la Teoría de Redes Neuronales en su publicación "The Organization of the Behaivor". Él fue el primero en investigar el aprendizaje psicológico y como se relacionaba con la actividad cerebral; concluyó que el aprendizaje se daba durante los cambios de los estímulos en las neuronas, proceso que es utilizado en una red neuronal artificial (Hebb, 1949).

En 1950 Karl Lashley descubrió que la información obtenida por el aprendizaje era almacenada de forma distribuida en la superficie cerebral.

Segfried plantea su sistema conocido como Pandemonium (1958), este sistema consta de una serie de capas compuestas por lo que se conoce como "demonios". En cada una de las diferentes capas de este sistema se reparten las tareas a realizar.

El perceptrón es la primera red neuronal, utilizada principalmente como identificador de patrones, fue diseñada por Frank Rosenblatt. Este tipo de red era capaz de reconocer patrones sin entrenamiento (1958).

La eliminación de eco en las líneas de teléfono, fue el primer problema real en solucionarse con una red neuronal bajo el modelo Adaline, el sistema fue puesto en marcha en 1960 por Bernard Widroff y Marcian Hoff.

Sin embargo en 1969, surge la polémica de que las redes neuronales no son fiables, en un estudio de prueba llamado "Perceptrons" de Marvin Minsky y Seymour Papert, lograron demostrar que un perceptrón no era capaz de resolver una simple función no lineal, lo que casi obligó a los investigadores de la época a desechar sus trabajos sobre redes neuronales.

En 1974, Paul Werbos desarrolló un algoritmo de aprendizaje conocido como Propagación hacia atrás.

La Teoría de la Resonancia Adaptada, en 1977, es una arquitectura de red que se diferencia de las demás por simular la memoria a largo y corto plazo.

John Hopfield, provocó el renacimiento de las redes neuronales con su libro "Computación neuronal de decisiones en problemas de optimización" (Hopfield 1982). Posteriormente en un trabajo de Rumelhart, Hinton y Williams desarrollaron el algoritmo de propagación hacia atrás para redes neuronales multicapa lo que dio nueva iniciativa al desarrollo del cómputo neuronal.

1.4 Computación digital vs computación analógica

En computación, existen dos tipos de perspectivas para resolver problemas. Cada una de estas perspectivas se diferencia una de la otra porque son complementarias, y cada una sirve para resolver problemas en específico que para su contraparte sería muy costoso.

El primer enfoque desciende del ábaco, de métodos mecánicos que usan sistemas de dígitos para procesar la información. Este tipo de computadoras son las digitales, se caracteriza por procesar datos exclusivamente numéricos o discretos.

El otro enfoque proporciona soluciones continuas en el tiempo. Surgió como un método de solución de problemas basado en compas y regla inventado por los agrimensores. Surge un método ingenioso al comparar los trazos con las propiedades que estas tenían. Las computadoras analógicas se basan en las relaciones análogas de una cantidad física asociada a una computadora y a las cantidades asociadas a un problema de estudio. La solución se obtiene al medir una cantidad continua de manera gráfica o en cambios de voltajes.

Una computadora digital se define como un dispositivo capaz de aceptar datos cuantitativos, los ordena y transforma mediante operadores matemáticos y lógicos, finalmente entrega resultados al usuario. Esta definición incluye a los dispositivos tales como el ábaco, reglas de cálculo, calculadoras, sistemas telefónicos y las computadoras personales.

La computadora analógica se define como una conexión entre dispositivos que obedecen a las relaciones matemáticas observables en el espacio continuo y que proporcionan datos que varían en el tiempo. Aunque la mayoría de las veces los resultados que entregan las computadoras analógicas son continuos, en algún momento los datos de entrada son procesados por sistemas digitales y en algún momento pasan por algún convertidor analógico-digital para ser procesados y registrados, además en la mayoría de los dispositivos analógicos pueden existir partes de naturaleza digital, lo que las convierte en computadoras híbridas.

1.5 Reconocimiento de patrones

Un patrón es un objeto que ya ha sido clasificado y puede usarse para clasificar otros objetos similares. El reconocimiento de patrones por lo tanto es clasificar los objetos similares de acuerdo a sus características.

En una definición más formal “un patrón es un conjunto de características que describen a una entidad u objeto a través de un conjunto de características que toman la forma de variables” (Devijver, 1982, p. 448).

Los tipos de patrones se distinguen por la cantidad, significado y naturaleza de estas características.

El proceso de clasificar y comparar las características de un objeto para obtener un patrón es relativamente sencillo. Cualquiera que sea el tipo de información para encontrar un patrón, ya sean imágenes, texto, sonidos, olores o personas; el proceso cognitivo sigue siendo el mismo, todo comienza desde la percepción.

Biológicamente hablando, todo comienza cuando percibimos un estímulo sensorial del ambiente, después de eso debemos recibir un estímulo semejante para que se pueda convertir en un patrón; este estímulo nuevo será un detonador de la memoria para que podamos asociar este nuevo estímulo con el anterior y finalmente establecer una correspondencia entre ambas sensaciones.

El reconocimiento de patrones mediante el uso de computadoras no es diferente a la parte biológica; con el uso de herramientas como sensores y convertidores analógico-digital una computadora es capaz de observar el ambiente continuamente, y almacenar la información recopilada que posteriormente puede ser usada para establecer relaciones entre los datos nuevos con los datos viejos y encontrar patrones y líneas de tendencia. Los datos pueden ser almacenados y consultados en cualquier momento, lo que es una ventaja en términos de velocidad y detalle de datos.

Un sistema de reconocimiento de patrones consta generalmente de 3 etapas (Devijver, 1982, p. 450):

1. Adquisición de datos
2. Extracción de características o parametrización
3. Clasificación de patrones

La adquisición de datos consiste en la obtención de datos directamente del objeto de estudio mediante algún sensor, algunas veces es necesario utilizar algún tipo de filtrado para facilitar la extracción de características (Salcedo, 2005, p.90).

La extracción de características es el proceso por el cual el sistema obtiene un conjunto reducido de magnitudes. La última etapa es aquella que clasifica los patrones mediante un clasificador. Los clasificadores necesitan ser entrenados antes de poder realizar el reconocimiento (Salcedo, 2005, p.91).

1.6 Procesamiento paralelo

Es un proceso empleado para acelerar el tiempo de ejecución de un programa dividiéndolo en múltiples trozos que se ejecutarán al mismo tiempo, cada uno en su propios procesadores (Gutiérrez A., 2009, cap. 4, p.4).

El uso de este tipo de procesamiento surge por la necesidad de evitar que diferentes programas interfieran entre sí evitando bloqueos mortales, cuellos de botella o invasión de páginas de memoria.

El principal objetivo del procesamiento paralelo es optimizar los tiempos de respuesta de los programas, sin embargo este mecanismo está limitado principalmente por la estructura del programa y por la arquitectura de la computadora.

Existen diferentes formas de paralelizar un programa para mejorar su rendimiento, siendo las más usadas por descomposición de datos, y la descomposición funcional.

La descomposición de datos; se usa cuando los datos a procesar son independientes entre sí y el orden en que sean manejados no afecte el resultado global del programa. Por ejemplo en la multiplicación de matrices.

Éste tipo de descomposición tiene una mejora de rendimiento lineal en cuanto al número de procesadores, a mayor número de procesadores es posible obtener una mejor respuesta; sin embargo tiene limitaciones muy graves a mayor número de procesadores significa mayor comunicación entre ellos lo que ralentiza y el número máximo de procesadores en que puede dividirse es el número máximo de operaciones independientes del problema.

La descomposición funcional; una aplicación consiste en varias tareas, cada tarea es responsable de una parte de la carga de procesamiento de la aplicación en general y a su vez, cada tarea realiza una operación independiente de las otras tareas (Gutiérrez A., 2009, cap. 4, p.4).

Niveles de paralelismo

	Ejemplos
Procesador	Procesadores vectoriales, división funcional
Multiprocesador	Memoria compartida y distribuida
Multicomputadoras	Clústers, bases de datos distribuidas

Tabla 1.2 Niveles de paralelismo

Características:

- Posee dos o más procesadores.
- Compartición de memoria global.
- Cada procesador dispone de su memoria caché para agilizar procesos.
- Compartición de dispositivos de entrada y salida, ya sea por bus común o por buses propios.
- El sistema operativo es el encargado de establecer la comunicación y el acceso a los recursos de lo procesadores.
- Procesamiento masivo de datos.
- Ganancia en el tiempo de respuesta.

Desventajas

- El tiempo de inicio puede ser mucho mayor al tiempo de procesamiento.
- Lectura de datos sucios, faltantes o fantasma, debido a la mala sincronización de los procesadores que acceden a los recursos compartidos.
- Sesgo en la división de los datos, la división no siempre exactamente igual para todos.

1.8 Campo de aplicación

Podría detallarse cientos de aplicaciones que surgen del análisis de las señales sonoras. El presente trabajo está enfocado a encontrar similitudes entre diferentes piezas de música para descubrir aquellas que tienen una relación ya sea por su género o por similitudes en su frecuencia. Las aplicaciones que pueden surgir son el desarrollo de nuevas piezas musicales, buscar contenidos de plagio, crear transductores de música, reconocedores y búsquedas de piezas.

En otros campos independientemente de la música, puede ser utilizado para realizar comparaciones médicas de diferentes enfermedades cardiacas, cerebrales, gastrointestinales y respiratorias; comparando los diferentes sonidos que emite el cuerpo con enfermedades ya establecidas y bien conocidas.

Otro ejemplo es la clasificación de diferentes familias de fauna, por lo diferentes sonidos que emiten al comunicarse y entender mejor su comportamiento.

Muchos son los campos beneficiados por el estudio del sonido; electrónica, medicina, biología, física, climatología, entre otros.

Algunos autores que han trabajado en proyectos similares y he tomado como referencia han sido: Durey y Clements que crearon un sistema capaz de detectar melodías usando modelos ocultos de Markov; Rabiner presento un trabajo cuyo sistema consistía en determinar cinco ritmos distintos; Beth Logan incorporo el uso de coeficientes cepstrales para el reconocimiento de características musicales. Francisco Salcedo busco la mejor configuración para aplicar técnicas de reconocimiento de voz a la música.

CAPÍTULO 2

PRINCIPIOS DEL SONIDO DIGITAL

Todo proceso consiste en un conjunto de entradas que ingresan a un sistema, pasan por una serie de operaciones y generan salidas. Las entradas son datos externos a la aplicación, pueden ser variables de captura, archivos o señales. La elección del tipo de entradas es un factor importante en un sistema; estas, en conjunto con las salidas definen la complejidad de las operaciones.

Los archivos de audio digital son fuentes de información ricas en datos, la mayoría solo legibles bajo ciertos estándares de codificación, otros con una sola lectura de las propiedades del archivo.

En este capítulo se intenta presentar un enfoque para el reconocimiento de las características importantes de los archivos de audio digital. Un archivo de audio digital es una estructura de datos que tiene la principal característica de almacenar información de una señal sonora que ha sufrido un proceso de digitalización, compresión y codificación.

Debido al tamaño de los archivos de audio es necesario extraer la información más importante de ellos. La razón es muy simple, al utilizar solo la información más importante el procesamiento de los datos se reduce mucho, sólo es necesario procesar el archivo de audio una sola vez para extraer su contenido y no cada vez que se requiera, el tamaño global se reduce hasta un 80% aproximadamente por no almacenar la información completa de cada archivo.

Los estándares de compresión actuales buscan dos cosas fundamentales; por un lado reducir al mínimo el tamaño de la señal digital y por otro conservar la mayor calidad posible. Los algoritmos de compresión más sencillos reducen muy poco con poco procesamiento. Los más complejos ofrecen una reducción mayor pero el procesamiento también es mucho más elaborado.

En cuanto a calidad, existen muchos parámetros para especificar el nivel de calidad de una señal sonora digital de una señal. Sin embargo la relación más aceptada para establecer este nivel consiste en una relación entre la señal de audio original y la señal de audio analógica, la exactitud de estas pruebas solo considera la sensibilidad humana. Para refinar estas pruebas, otro método consiste en determinar la relación de señal a ruido donde la diferencia es más exacta que para señales idénticas para el oído humano.

2.1 Acústica

La acústica es la ciencia "que estudia las ondas sonoras en su generación, propagación y recepción, así como las circunstancias que producen estos tres factores cuando se crea una perturbación acústica" (Ref. 24, Pohlmann, 2002, p1).

Una definición más precisa del sonido, nos la da Serdi (1997):

> Lo que entendemos por sonido es fruto de una compleja interacción entre un objeto vibrante, un medio transmisor (frecuentemente el aire), el oído, y el cerebro. Para que la vibración sea audible para un ser humano, este objeto debe oscilar aproximadamente entre 20 y 20.000 veces por segundo. Al oscilar, el objeto desplaza el aire que lo rodea, comprimiendo y descomprimiendo periódicamente las moléculas que lo integran, y modificando por consiguiente la presión del aire de forma periódica. Dado que las moléculas desplazadas van empujando a las contiguas, la variación periódica de la presión se propaga originando lo que recibe el nombre de ondas sonoras. Cuando las ondas llegan al oído, el cerebro interpreta estas variaciones de presión como sonido. (p. 9).

El sonido se mide considerando las vibraciones de las partículas del aire, cuándo estas partículas se comprimen y descomprimen generan pequeñas variaciones de presión. Las unidades de presión son demasiado pequeñas como para considerarlas como su unidad fundamental, para evitar cálculos ínfimos la unidad designada son los decibelios.

$$(\text{dB}) = 20 \log_{10} (\text{Presión}/\text{Presión de referencia})$$

Esta fórmula permite calcular valores para los niveles de presión sonora estableciendo una relación entre la presión de las ondas y la presión mínima perceptible por el oído humano.

Aunque el sonido es capaz de propagarse en cualquier medio de físico, la velocidad de propagación depende principalmente de la densidad del medio, entre mayor sea la densidad mayor será la velocidad de propagación. Considerando esto podemos darnos cuenta que la velocidad en el aire es menor que la velocidad en el agua, y ésta a su vez es menor que en los metales.

Medio de transmisión	**Velocidad aproximada del sonido A temperatura ambiente**
Aire	343 [m/s]
Agua	1500 [m/s]

Metales	5000 [m/s]

Tabla 2.1 Velocidad del sonido en distintos medios de transmisión.

Sin embargo estas velocidades son relativas, solo se considera un medio constante sin disturbios, bajo ciertos parámetros. Los fenómenos que permiten establecer medidas en la perdida de energía en una señal acústica son la reflexión, absorción y la difracción.

La reflexión "se produce cuando una onda choca contra una superficie de otro medio, esto provoca que la onda original se divida en dos señales una de reflexión y otra de transmisión dividendo entre ambas la energía de la señal original" (IPM, 2000, p.8). La proporción entre la señal de reflexión y la señal de transmisión depende del ángulo de inclinación entre los medios y del medio.

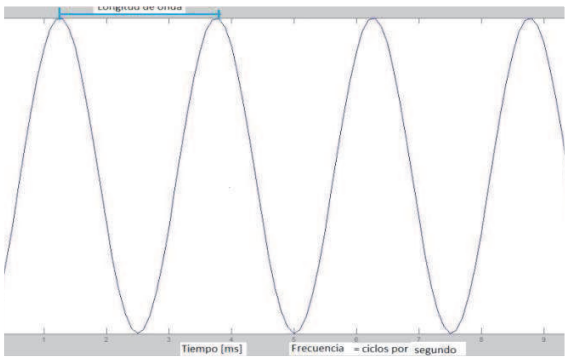
La absorción "es producida por la fricción de la señal con el medio de transmisión, el roce constante libera energía en forma de calor que es sustraída de la señal" (IPM, 2000, p.8). La cantidad de la pérdida de energía depende de la frecuencia de la señal siendo mayor para altas frecuencias que para las bajas frecuencias.

La difracción "se produce cuando la trayectoria de la señal es interrumpida por diferentes obstáculos que se encuentran en el medio, la presencia de los obstáculos disminuye la intensidad de los sonidos que pasan a través de ellos" (IPM, 2000, p.9).

2.1.1 Sonidos periódicos

Los sonidos rara vez son producidos por una única perturbación, Miyara (1999) explica que "los sonidos de la naturaleza son el resultado de múltiples perturbaciones sucesivas. Estos sonidos se denominan periódicos, y pueden dividirse en ciclos, donde cada ciclo abarca todo lo que sucede entre dos perturbaciones sucesivas en el aire" (p.4).

Una vez habiendo definido una onda periódica es necesario definir los parámetros que conforman una onda periódica. La figura 2.1 muestra gráficamente las componentes de una señal periódica.



Fígura 2.1 Ejemplo de una señal periódica.

La longitud de onda λ “es la distancia entre dos perturbaciones sucesivas en el espacio. Se mide en metros [m], centímetros [cm]” (Miyara, 1999, p.5).

La frecuencia *f* es de “una señal es el número de ondas que pasan por un punto durante un segundo” (Tippens, 2002, p.472). Su unidad de medida es el Hz.

$$1Hz = 1\ ciclo = \frac{1}{s}$$

El periodo *T* es el tiempo transcurrido entre una perturbación y otra, se mide en segundos.

$$T = \frac{1}{f}$$

La siguiente tabla muestra los rangos de parámetros para los sonidos audibles:

	Sonidos Agudos	**Sonidos graves**
Longitud de onda	2[cm]	17 [m]
Frecuencia	0.05[ms]	50[ms]
Periodo	20[kHz]	20[Hz]

Tabla 2.2 Limites del sonido audible

La amplitud “se define como el valor máximo que alcanza una oscilación en un ciclo. También se denomina valor pico” (Tippens, (2002), p. 473).

“La fase indica la posición de la partícula que oscila en el momento de empezar a contar el tiempo *t,* es decir $t = 0[s]$. La fase se mide en radianes (rad) o en grados (^{o})” (IPM, 2000, p. 11).

2.1.2 Envolvente

La amplitud de un sonido no necesariamente debe ser constante, sino que puede variar en el tiempo, como se observa en la figura 2.2. Es más, la mayoría de los sonidos poseen una amplitud variable. Se llama envolvente, “a la forma que se obtiene uniendo las amplitudes de los ciclos sucesivos” (Miyara, 1999, p. 9).

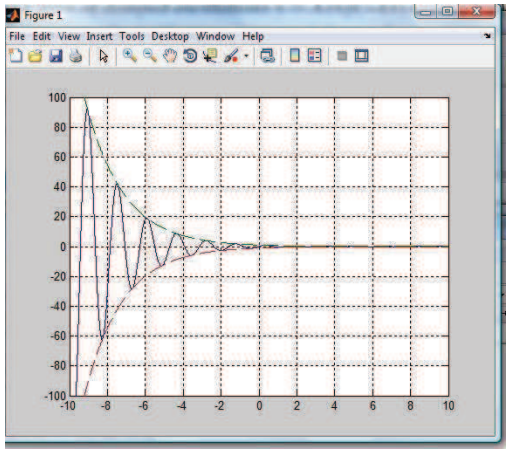


Fig. 2.2 Envolvente de una señal.

2.1.3 Espectro

Cualquier sonido periódico puede representarse como una serie de frecuencias características, estas frecuencias características son el resultado de la descomposición de una señal. La serie completa:

$$f_n = nf_1 \qquad \text{Donde } n = 1,2,3 \ldots$$

Ésta conformada por la frecuencia fundamental y sus sobre tonos, y se conoce como serie armónica. La fundamental es la primera armónica y el primer sobre tono es la segunda armónica” (Tippens, (2002), p.479).

La información sobre las frecuencias que contiene un determinado sonido y sus respectivas amplitudes constituyen al espectro. “El espectro se obtiene calculando la energía que aporta a cada frecuencia al sonido total” (IPM, 2000, p.11).

El espectro se puede representar de dos maneras, la primera mediante una tabla que contiene el número del armónico y la amplitud de la onda; la segunda es mediante una herramienta grafica llamada espectrograma, el cual es una gráfica que contiene en el eje horizontal la frecuencia y en el vertical la amplitud para cada armónico.

Para sonidos no periódicos existen los denominados espectros inarmónicos, en el cual no es posible identificar la frecuencia ni el periodo. “En el caso de los espectros inarmónicos también pueden existir una variación en el tiempo, pudiendo en este caso inclusive variar no sólo la amplitud de los sonidos parciales sino también la frecuencia” (Miyara, 1999, 16).

Existe otro tipo de sonidos. El ruido es una cantidad de sonidos parciales muy grande muy próximos entre sí. Algunos ejemplos son el sonido del mar, conversaciones, sonido ambiental, etc. “La forma de representar el espectro es mediante una curva continua llamada densidad espectral” (Miyara, 1999, p.17).

Existen dos tipos de ruido. El ruido blanco se caracteriza por tener una densidad espectral constante. “El término blanco proviene de una analogía con la luz blanca, que contiene todos los colores del espectro con la misma intensidad” (Miyara, 1999, p.18).

El ruido rosa contiene mayor proporción de bajas frecuencias (contiene todas las frecuencias pero las más bajas corresponden al color rojo) tiene la particularidad de que en cada octava (el intervalo de frecuencia entre un do y un re) tiene la misma energía sonora.

2.2 Psicoacústica

La psicoacústica, "es la ciencia encargada del estudio de la percepción subjetiva de las cualidades del sonido. Éstas son intensidad, tono y timbre. Dichas cualidades o características, están a su vez determinadas por los propios parámetros del sonido, sobretodo la amplitud y la frecuencia" (Arribas J.I, 2006).

- Sensaciones psicoacústicas

"Cuando escuchamos un sonido, percibimos sensaciones que pueden ser clasificadas en tres tipos: la altura, la intensidad y el timbre." (Miyara, 1999, p. 18)

2.2.1 Altura

"La altura nos permite distinguir un sonido agudo, de un sonido grave y depende del número de vibraciones por segundo que ejecuta un cuerpo sonoro" (Torres, 1972, p.15).

Esta cualidad es puramente subjetiva, sin embargo existe una relación física entre la altura y la frecuencia de la vibración que la produce. Para que el sonido tenga una altura definida es necesario que la frecuencia de oscilación sea aproximadamente periódica. El periodo de tiempo para percibir la altura no necesariamente debe ser largo, en tiempos largos podemos considerar la altura como una función del tiempo.

Una forma más sencilla de representar las alturas es mediante las octavas, la frecuencia de un sonido determina la altura del mismo, una relación de proporcionalidad en la frecuencia representa un cambio de altura, esto es lo que se conoce como octava.

Para simplificar la construcción de instrumentos, se acordó dividir la octava en doce alturas, de las cuales siete son conocidas como las notas musicales (do, re mi, fa sol, la, si) y las restantes toman el nombre de la inmediata anterior agregándole el símbolo b (bemol). La relación entre las frecuencias de una nota a la siguiente siempre vale $2^{1/12}$. De esta manera, al avanzar las doce alturas, obtenemos un valor de exactamente 2.

El análisis de bandas de octavas ha sido definido como un estándar para el análisis acústico.

2.2.2 Timbre

"El timbre es la cualidad por la cual distinguimos un instrumento de otro, una voz de otra. Depende de los sonidos armónicos, que también se llaman secundarios o concomitantes" (Torres, 1972, p.15).

Diferentes sonidos pueden tener la misma altura pero no suenan igual, esto se debe a que en los sonidos la frecuencia de más grave es la que determina el periodo y la altura. La frecuencia más grave se llama frecuencia fundamental, las otras frecuencias son múltiplos de la frecuencia fundamental y se llaman armónicos.

La suma de los diferentes armónicos determina el timbre de la señal, y la identifican de las demás.

Jean Baptiste Fourier, fue el primer matemático que emprendió el estudio de sobre los armónicos, el descubrió que una señal por compleja que sea puede descomponerse en una suma algebraica de señales sinusoidales armónicas, obtenidas de una original.

Los factores que influyen en la estructura armónica son los números, magnitud y fluctuación de los armónicos, junto con la presencia o ausencia de armónicos superiores; al ancho de banda de la señal y la energía aportada a la misma por los armónicos en relación con la energía total.

Miyara Fernández, 1999, establece dos enfoques para el análisis del timbre:

> El primer enfoque estudia los sonidos aislados, y se propone identificar todos los elementos que los distinguen de otros sonidos, intervienen dos elementos: el espectro y las envolventes. Hay una envolvente primaria, que es la que determina la forma en que varía en el tiempo la amplitud general, y una serie de envolventes secundarias, que corresponden a las variaciones temporales relativas de los armónicos o de los parciales (p. 22 – 23).

El segundo enfoque clasifica los sonidos según la fuente:

> Busca las características comunes a todos los sonidos de un instrumento o de una voz, y las que los distinguen de los sonidos de otros instrumentos o voces. El elemento fundamental de este análisis es la existencia de resonancias en los componentes accesorios al mecanismo propiamente dicho de producción del sonido, resonancias que filtran el sonido, favoreciendo determinadas frecuencias más que otras (p. 24).

2.2.3 Intensidad

"La intensidad nos permite distinguir un sonido fuerte de un sonido suave y depende de la amplitud de las vibraciones del cuerpo" (Torres, 1972, p. 15). El análisis de la intensidad es más complejo que el de la altura, el orden de la intensidad de un sonido es muy variable y se trata de un crecimiento logarítmico en las amplitudes de la señal. La intensidad se mide en decibelios.

2.2.4 Fisiología

El sistema auditivo es el sistema que nos permite percibir los oídos. La fugura 2.3 representa las componentes fisiológicas del oído humano

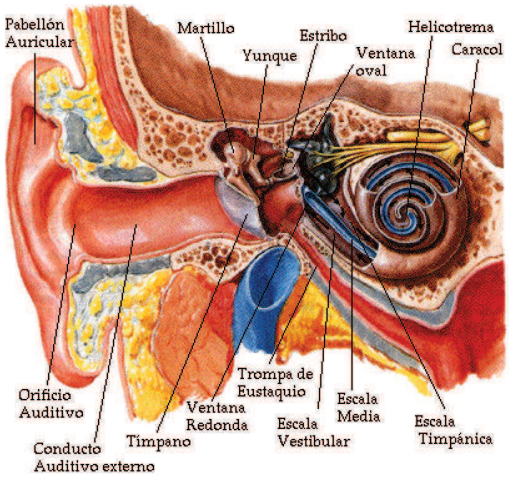


Fig. 2.3 Anatomía interna del oído humano.

Su estructura se compone de tres partes principales:

Oído externo: “Diseñado estructuralmente para recoger las ondas sonoras y dirigirlas al interior durante el proceso de audición” (Tresguerres, 2009, p. 74).

Oído medio: “Transforma la energía acústica en energía mecánica transmitiéndola y amplificándola hasta el oído interno” (Tresguerres, 2009, p. 74).

Oído interno: “Se realiza la transformación de la energía mecánica, producida por las ondas sonoras, en energía eléctrica” (Tresguerres, 2009, p. 74).

2.2.5 Direccionalidad

“La direccionalidad se refiere a la capacidad de localizar la dirección de donde proviene el sonido” (Miyara, 1999, p.26).

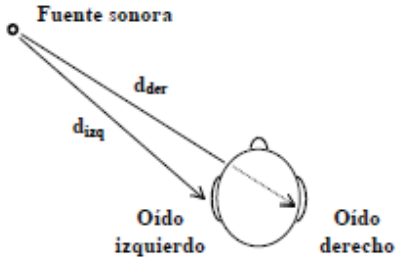


Fig. 2.4 Diferencia de intensidad de una fuente hacia los oídos.

La forma en la que el cerebro puede localizar la fuente está relacionada con las diferencias percibidas por cada oído: por un lado ésta la diferencia de intensidad entre las señales recibidas por cada oído debido a que la cabeza produce una difracción del sonido: y la segunda es una diferencia de fase producida por la distancia entre ambos oídos, como se aprecia en la figura 2.4.

En un esquema enfocado a la psicoacústica, para determinar la direccionalidad es necesario tomar en cuenta tres factores:

1. El retardo temporal. “Tiene que ver con la diferencia de fase y se debe a que un mismo sonido producido por la misma fuente sonora casi nunca es igual para un oído que para el otro.” (Arribas I., 2006)
2. El efecto Haas: “si varios sonidos independientes llegan a nuestro cerebro en una ventana temporal inferior a 50 ms, éste los fusiona y los interpreta como uno sólo”
3. Enmascaramiento. Es una cualidad que permite a unos sonidos ocultar otros sonidos, para hacerlos imperceptibles El enmascaramiento es una propiedad del oído y no del sonido.

2.2.6 Espacialidad

La espacialidad, “nos permite asociar un sonido con el ambiente en el cuál éste se propaga, estimar las dimensiones de una habitación o una sala sin necesidad de recurrir a la vista” (Miyara, 1999, p. 26).

Intervienen varios factores:

La distancia entre la fuente y el oído. Las reflexiones tempranas que proveen al a oído y una noción de la dimensión de la habitación creando una sensación de ambiencia. La reverberación se produce por la superposición de las reflexiones tardías, estas son las reflexiones de las reflexiones. La figura 2.5 representa como el intervalo de tiempo entre cada reflexión disminuye.

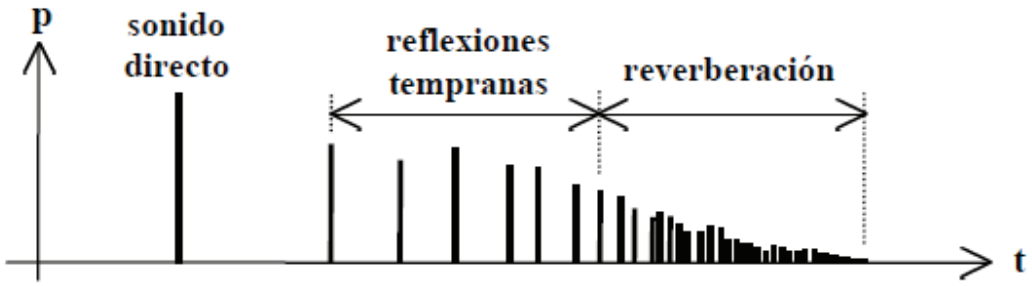


Fig. 2.5 Propagación del sonido en un espacio semi-abierto. La intensidad y los intervalos de tiempo disminuyen a cada reflexión.

Finalmente, está el efecto Doppler, se produce por fuentes en movimiento. Si la fuente del sonido se acerca a nosotros lo escuchamos más agudo, pero si se aleja la altura disminuye haciéndose más grave.

2.3 Análisis del audio digital

El audio digital es un proceso de transformación de señales acústicas a un conjunto de información binaria, mediante técnicas de conversión analógica-digital.

Citando a Pohlmann, 2002:

> El audio es analógico en su origen y, por lo tanto, los sistemas digitales deben transformar este carácter analógico, mediante los procesos de muestreo y codificación… Las muestras de una señal de audio deben cumplir el teorema de muestreo,… el teorema establece una condición: la señal debe ser de banda limitada. Además el muestreo debe realizarse con cierta precaución, ya que puede producir un aliasing. El error de cuantificación se puede minimizar aplicando técnicas de dithering (p.19).

El sistema de conversión completo tiene dos conversores, un conversor analógico digital a la entrada y un conversor digital-analógico a la salida. El proceso comienza cuando la señal atraviesa el convertidor, durante el proceso la señal de muestrea, cuantifica y codifica, otros procesos más complejos utilizan filtros para eliminar frecuencias indeseables, o eliminar ruido. La figura 2.6 muestra un diagrama con las etapas del proceso A/D:

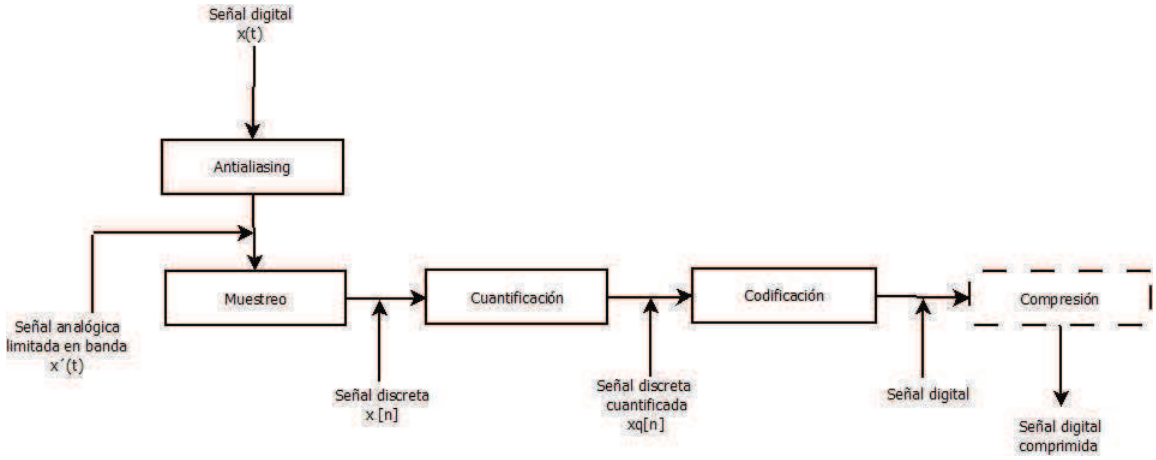


Fig. 2.6 Proceso de conversión de una señal analógica a una señal digital.

El muestreo toma pequeñas porciones de datos de la señal original en función a la señal del reloj del sistema, la cuantificación discretiza las muestras, y la codificación asigna valores binarios a cada valor discreto. Las tasas de muestreo y el número de bits por muestra determinan la calidad del audio. El proceso inverso recupera parte de la señal original.

2.3.1 Teorema del muestreo

El teorema de muestreo establece que “una señal continua limitada en banda puede ser remplazada por una secuencia discreta de muestra sin pérdida de información, y describe cómo se pueden reconstruir la señal original a partir de esas muestras” (Pohlmann, 2002, p.21).

> “El teorema especifica que la frecuencia de muestreo debe ser al menos el doble de la frecuencia máxima original. Las señales de audio con frecuencias entre 0 y la frecuencia de Nyquist (S/2) Hz pueden especificarse exactamente con S muestras por segundo” (Nyquist-Shannon, 1949).

El tratamiento para señales de audio es el siguiente:

Primero las señales se pasan por un filtro pasa-bajas, para que la respuesta en frecuencia quede limitada en banda y no exceda la frecuencia de Nyquist, además el filtro también elimina aquellas frecuencias que no son percibidas por el oído. Puede verse en la figura 2.7 una señal que ha sido preparada para la conversión.

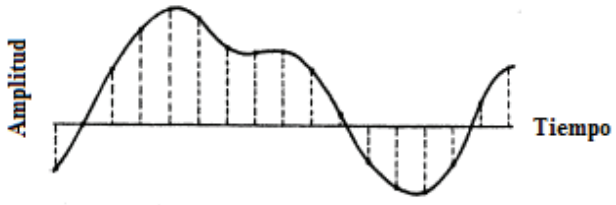


Fig. 2.7 Señal analógica filtrada.

La señal se muestrea para obtener valores en amplitud, estos valores representan impulsos de la señal. La señal muestreada contiene la misma información que la señal a la salida del filtro. En la figura 2.8 se observan amplitudes de la onda tras el muestreo.

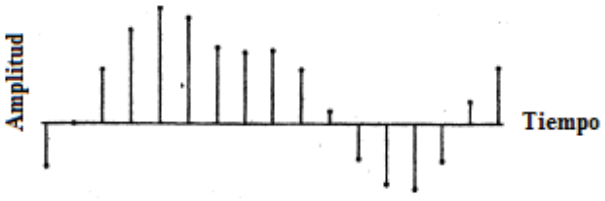


Fig. 2.8 Señal analógica muestreada.

Posteriormente, la señal puede ser reconstruida a su forma original, sin ningún tipo de pérdida de información. La figura 2.9 da una visión simple pero coherente del proceso de reconstrucción.

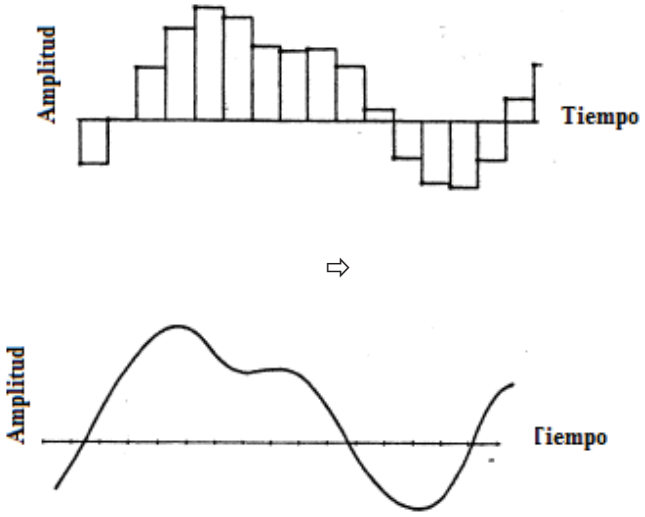


Fig. 2.9 Reconstrucción de una señal analógica de una señal digital.

Una señal muy variable requiere una tasa de muestreo mucho más alta esto requiere un proceso de cuantificación más intenso, la selección de una frecuencia de muestreo es muy importante porque determina el ancho de banda del sistema.

El teorema de muestreo especifica cómo debe muestrearse una señal para asegurar un determinado ancho de banda. “La frecuencia de muestreo debe ser por los menos el doble de la frecuencia máxima de la señal de audio para tener un muestreo sin pérdida de información” (Pohlmann, 2002, p.22). El uso de un filtro asegura que las frecuencias superiores sean eliminadas para evitar al aliasing. Y para asegurar la recuperación de la señal original también es necesario colocar otro filtro paso-bajas para eliminar la distorsión de armónicos totales.

La finalidad de elegir un ancho de banda que nos favorezca viene determinada por la necesidad de modificar la frecuencia de muestreo para ajustar el tamaño. La frecuencia de muestreo también determina las características de muestreo de los equipos electrónicos así como la transmisión y el tamaño de almacenamiento de la señal. A frecuencias muy altas la señal requiere un procesamiento mucho más veloz, así como un espacio para su almacenamiento más grande y un canal de transmisión mayor para obtener el mismo rendimiento.

2.3.2 Aliasing

El aliasing es un fenómeno anómalo que aparece en el proceso de muestreo. El aliasing puede crear componentes falsas en una señal, estas componentes aparecen dentro del ancho de banda de la señal y son indetectables. Durante el proceso de muestreo sin pérdidas, la condición que se debe cumplir es tener un ancho de banda limitado. “El aliasing aparece cuando se viola el teorema de muestreo. La frecuencia más alta de la señal de audio debe ser igual o inferior a la frecuencia de Nyquist” (Pohlmann, 2002, p. 25).

Cuando las frecuencias alcanzan la frecuencia de Nyquist, se crean dos muestras por ciclo. Con frecuencias más altas, el proceso continúa creando muestras pero con información falsa de la señal. Los componentes de aliasing no solo ocurren alrededor de la frecuencia de muestreo sino también en múltiplos de la misma.

La forma de resolver este problema es empleando un filtro paso-bajas como ya se había mencionado antes, el filtro debe proporcionar una atenuación a partir de la frecuencia de Nyquist para asegurar que el espectro de la señal no tenga frecuencias superiores. Sí el filtro no es capaz de corregir el aliasing, una vez se muestre la señal no habrá forma de eliminar las componentes erróneas en la señal digital.

2.3.3 Cuantificación

“El muestreo representa los instantes de medida, y la cuantificación representa los valores de medida, en audio representa la amplitud de la señal en los instantes de muestreo” (Pohlmann, 2002, p.26). Una señal analógica puede representarse mediante una serie de pulsos, la amplitud de cada pulso indica el valor numérico de la señal en ese preciso instante. La precisión del valor de la cuantificación está determinada por la resolución del sistema. La resolución del sistema depende de la cantidad de bits utilizados para representar la amplitud.

En una cuantificación uniforme, la amplitud de la señal se transforma en un determinado número de niveles de cuantificación, todos de igual tamaño, para poder precisar lo más posible el valor original. Los sistemas de audio de alta calidad utilizan al menos 65 000 niveles de cuantificación.

“La cuantificación el proceso en el cual a cada muestra se le asigna un valor de un conjunto finito de niveles” (Martínez, 2009, p.65) .Teóricamente el muestreo es un proceso que no produce pérdidas de información, por el contrario la cuantificación, no importa cuál sea la escala ni cuántos sean los niveles de cuantificación, o el código utilizado siempre existe un error.

2.3.4 Dither

"El dither es una técnica empleada en audio digital para eliminar la distorsión producida por el proceso de cuantificación. Su misión es transformar la distorsión en ruido blanco" (Martínez, 2009, p.88). El dither es un ruido de bajo nivel incorrelado con la señal de ruido, se añade a la señal de audio antes de ser muestreada. Cuando se añade el dither la amplitud de la señal cuantificada se balancea en torno a los niveles de cuantificación.

Para Pohlmann, el dither "no enmascara el error de cuantificación; más bien permite al sistema de digitalización codificar amplitudes inferiores al bit menos significativo" (p. 35).

2.3.5 Codificación

Martínez (2009) establece que "la codificación es la etapa en la cual, una vez se ha obtenido un conjunto discreto de muestras cuantificadas, son asignadas a símbolos o valores que las representan" (p. 66).

Con el objetivo de facilitar el trabajo de la codificación se ha establecido la numeración binaria como método de asignación de valores a las muestras, principalmente por el hecho de que los niveles de cuantificación siempre son potencias de 2 (2^n).

2.3.6 Compresión

Estrictamente, la compresión de la señal resultante no es un proceso de la conversión analógica-digital, debido a que la señal digital ya fue obtenida en la codificación; sin embargo es prudente mencionar este proceso porque muchas de las aplicaciones que requieren una conversión analógica – digital o viceversa han sometido a las señales a un proceso de compresión de información.

Existen dos tipos de compresión:

Compresión sin pérdidas: "La información digital que se recupera es exactamente la misma que la que se introduce en un sistema de compresión" (Martínez, 2009, p. 67).

Compresión con pérdidas: "Se desprecia parte de la información de la señal original para reducir el volumen ocupado por la misma, por lo que, en el receptor, es imposible reconstruir la señal de forma exacta, se pierde información de manera irreversible" (Martínez, 2009, p.67).

Para las señales digitales de audio, el método más difundido en la actualidad es el mp3. “Los sistemas de compresión analizan el espectro de la señal de audio, obteniendo una distribución en tonos y bandas para después, aplicando métodos psicoacústicos, eliminan el contenido que el oído no puede percibir” (Martínez, 2009, p. 68).

En resumen, el muestreo y la cuantificación son los dos procesos fundamentales de la digitalización. El muestreo determina el ancho de banda de la señal y por lo tanto la respuesta en frecuencia. La cuantificación determina el rango dinámico del sistema, que puede ser medido como la relación señal/ruido.

2.4 Herramientas de análisis de los parámetros del sonido

Las herramientas de análisis son instrumentos de medición que permiten el estudio de las señales sonoras

2.4.1 Medidores de intensidad

El nivel máximo de intensidad es de 0dB. El nivel pico de nivel de intensidad es la amplitud más alta alcanzada en un instante dado. La forma de obtener los valores de intensidad sonora es mediante el uso del nivel RMS que se aproxima mucho al nivel percibido por el nivel del oído humano.

El nivel de RMS (Root Mean Square) corresponde una media cuadrática. Para calcularlo con valores discretos se usa la siguiente formula.

$$x_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$$

Donde x son los valores de amplitud para diferentes tiempos.

Para una función continua *f(t),* en un intervalo dado la fórmula para calcular el RMS es la siguiente:

$$x_{RMS} = \sqrt{\frac{1}{T_2 - T_1}\int_{T_1}^{T_2} [f(t)]^2 dt}$$

2.4.2 Analizadores de espectro

El espectro de un sonido es la información sobre la distribución de las frecuencias que lo componen y sus respectivas amplitudes. La descomposición de las frecuencias se realiza a través de la transformada de Fourier. Existen tres familias de herramientas para el análisis espectral: los espectrogramas, los FFT y los analizadores por bandas.

2.4.3 Espectrogramas

“Nos dan una imagen de la densidad espectral de una señal en el tiempo, pero no utilizando números, sino colores” (San Martín, 2012, p. 4). A medida que transcurre tiempo en una señal, va dejando una huella de color para cada rango de frecuencia y su correspondiente intensidad. La función principal del espectrograma es poder hacer comparaciones dentro del espectro de una señal por intervalos de tiempo mediante variaciones en los colores.

2.4.4 FFT (Fast Fourier Transform)

Los FFT solo muestran la composición espectral de un instante en específico. “Son herramientas muy útiles y precisas a altas frecuencias, ya que permiten hacer una partición del espectro en bandas equidistantes.... Por el contrario para bajas frecuencias se usa interpolación para obtener valores más exactos” (San Martín, 2012, p.5). Si se utilizaran grandes tiempos de integración para señales como una pieza musical, no se podrá hacer un análisis espectral en tiempo real.

2.4.5 Medidores de bit

Es un medidor de ancho de banda, y sirve para ver la profundidad de bits que se está utilizando.

- LM5 Y LM5D

“Es un medidor estandarizado para EBU e ITU, el cual funciona en forma de radar, permitiendo observar la evolución temporal del loudness” (San Martín, 2012, p.10).

Otras características:

- ✓ Posee un medidor pico
- ✓ Contiene descriptores estadísticos como Centro de Gravedad (Volumen promedio)
- ✓ Permite visualizar la coherencia o variaciones de intensidad
- ✓ Indicadores de distorsión

2.5 Calidad de música digital

La calidad determina una comparación entre la señal original y la señal codificada.

2.5.1 Respuesta en frecuencia

"La respuesta en frecuencia determina el comportamiento de la frecuencia de una señal de un dispositivo en relación a las frecuencias que integran el espectro de audio" (IPM, 2000, p.17). En otras palabras, todos los dispositivos incluyendo al oído son capaces de procesar diferentes frecuencias pero eso no significa que tengan una respuesta igual para todas las frecuencias. La respuesta en frecuencia de cualquier sistema debería ser plana, lo que significa que el sistema trata igual a todo las señales entrantes, con que tiene la misma respuesta.

Por otra parte cada dispositivo tiene una respuesta en frecuencia característica. En un sistema mayor, donde una señal pasa a través de un conjunto de equipos conectados de forma que la salida es la entrada de otro, la respuesta en frecuencia de cada dispositivo se suma a la anterior hasta obtener una respuesta en frecuencia total. La importancia de esto recae en la calidad individual de cada equipo donde uno puede ser el causante de que una señal no se escuche bien.

De acuerdo a lo anterior, calidad en un sistema de audio depende principalmente de los equipos electromecánicos, principalmente los altavoces quienes son los principales protagonistas de la calidad del sistema. "Otra filosofía muy acertada a la hora de buscar el mejor sonido a base de no modificar el espectro, es desechar todas las etapas en la cadena de sonido que no sean necesarias; ya que por muy calidad que tengan, siempre alteran la señal" (IPM, 2000, 19).

2.5.2 Distorsión

La distorsión es la medida más usada para realizar un análisis de calidad, en dispositivos de audio este parámetro viene especificado por el fabricante.

Existen tres tipos de distorsión.

<u>Distorsión de amplitud.</u>

"Se produce cuando una señal salida no guarda la relación de amplitud en diferentes frecuencias que la señal de entrada" (IPM, 2000, p.19). Un ejemplo de este tipo de distorsión se da en la respuesta de en frecuencia.

Un ejemplo más generalizado se produce cuando un amplificador intenta maximizar una señal por encima de los valores del dispositivo. El dispositivo posee

un valor de voltaje máximo a su salida, el amplificador incrementa los picos de la señal a valores tan grandes que el dispositivo se ve forzado a recortar estos picos para que la salida sea la misma a la especificada.

Distorsión de fase

Se produce cuando "la salida no conserva la relación de fase entre las diferentes frecuencias de entrada" (IPM, 2000, p. 20). Este tipo de distorsión simplemente se ignora, la razón es porque el oído humano no es capaz de percibir una diferencia de desfasamiento.

Distorsión de Armónicos Totales o THD

La THD se produce cuando en la salida de una señal aparecen armónicos no deseados. Cuanto mayor es la proporción de los armónicos respecto al original el efecto que tendrá será peor en el sonido.

El THD se mide con un porcentaje, y por lo general nunca superan el 1%. "El porcentaje representa la parte del total de la energía a la salida, que pertenece a los armónicos, en otras palabras a la distorsión" (IPM, 2000, p.21). La fórmula es:

$$THD\% = 100\sqrt{\frac{v1^2 + v2^2 + v3^2 + \cdots}{v0^2}}$$

Los valores de v1, v2, etc. representas la amplitud de los armónicos, mientras que v0 es la amplitud original de la frecuencia.

Distorsión de intermodulación o IMD

La IMD se produce cuando las diferentes frecuencias de una señal crean nuevas frecuencias, Las frecuencias nuevas siempre aparecen en función a la frecuencia más alta y separadas de ella en múltiplos de la frecuencia más baja.

La distorsión de intermodulación se mide en porcentaje (%), y se calcularía midiendo la tensión de las frecuencias de intermodulación y aplicando la siguiente fórmula:

$$IMD\ \% = 100\sqrt{\frac{\sum(vi)^2}{vo^2}}$$

Donde vi son las amplitudes de las frecuencias resultantes y vo es el voltaje de la frecuencia mayor de la señal de entrada.

2.5.3 Relación señal/ruido

La relación señal a ruido es una diferencia en dB entre una señal y otra señal introducida por el dispositivo o alteraciones electromagnéticas del ambiente. El ruido es una señal aleatoria que afecta la señal original provocando distorsiones. La relación señal a ruido y se calcula midiendo una señal normalmente en un dispositivo y comparándola con la misma medición sin la señal original dejando solo el ruido. La existencia de ruido es inevitable, los equipos más sofisticados solo pueden disminuir el nivel de ruido los más insignificantemente posible pero no eliminarla.

2.5.4 Diafonías

O “separación de canales” (IPM, 2000, p.23), una diafonía es una separación de canales, se produce cuando a la salida de un canal se sobrepone la entrada de otro canal. Las diafonías son directamente proporcionales a las frecuencias, esto es porque al aumentar las frecuencias la separación entre los canales disminuye.

CAPÍTULO 3

ANÁLISIS MUSICAL

“La música es el arte de combinar bien los sonidos en el tiempo” (Torres, 1972, p. 13).

El arte es el concepto que engloba todas las creaciones realizadas por el hombre para expresar una visión sensible acerca del mundo, ya sea real o imaginario. Mediante recursos plásticos, lingüísticos o sonoros, el arte permite expresar ideas, emociones, percepciones y sensaciones.

La música realiza una clasificación de los sonidos, en notas que a su vez se clasifican en escalas. Físicamente cada nota se puede asociar a una frecuencia fundamental y una serie de armónicos que varían en función del instrumento.

Un conjunto de notas conforman un compás, la definición musical de tiempo se define como cada una de las partes de igual duración en las que se divide el compás.

El sistema debe ser capaz de determinar y calcular las estructuras musicales más simples como son las notas, duración y compás.

3.1 Problemas de clasificación musical

La música es una compañera que ha existido desde los orígenes del hombre, y sin embargo no ha existido ningún inconveniente en clasificar una pieza musical; en su forma más simple nos gusta o no. Tampoco implica un mayor problema el clasificarlos según el instrumento, el ritmo, la melodía, el estilo o por cómo nos hacen sentir.

La computadora no es capaz de realizar una clasificación global de la música ni mucho menos discernir las características fundamentales de la música; sin que se les haya tratado apropiadamente. El proceso de trasformación de la música como una señal ya se ha tratado anteriormente.

a) La señal musical es continua, no existe un lineamiento para poder separar los conjuntos de notas.
b) La música es variable, a pesar de que existen pautas y mecanismos para mantener el orden en una composición musical; la cantidad de instrumentos y de voces interfieren en la obtención de un análisis coherente.
c) No existen métodos capaces de separar las diferentes señales de los instrumentos y de las voces para obtener un análisis confiable de cada una de ellas.
d) A pesar de que existe una ciencia que estudia la estructura musical desde su forma más simple (nota) hasta la más compleja, la capacidad de cómputo es tan demandante que no es posible convertir esa estructura en una gramática fiable.

Eric D. Sheirer (2000), en su trabajo de investigación propone seis formas de clasificar los patrones en la música:

a) Detección del ritmo. Imitan la capacidad humana de detectar la periocidad de la música. Implica el conocimiento de la estructura musical junto con la percepción humana.
b) Reconocedor de instrumentos. Tratan de identificar los diferentes instrumentos que intervienen en una pieza musical, basan su funcionamiento en la gama característica de los armónicos de cada instrumento.
c) Detección de melodía. El objetivo es extraer la melodía básica de una pieza musical y realizar una transcripción para compararlas con otras.
d) Identificación de estilo musical. Identifican el estilo musical y los clasifican.
e) Detección de sensaciones emocionales. Imitan la capacidad humana de clasificar la pieza de acuerdo a la apreciación que transmiten.
f) Reconocimiento general. Permiten la detección de varias características musicales a la vez.

3.2 Notas musicales

"El Solfeo es la disciplina perteneciente a la música que estudia la teoría y la práctica del uso de los signos musicales" (Seguí, 1984).

Una nota musical se define estrictamente por tres valores: su nombre, su escala y su símbolo.

1. El nombre, define a cada una de las notas musicales de la más grave a la más aguda. Son siete y cada una de ellas representa todos los tipos de sonidos reproducibles por un instrumento o por la voz humana. "Los nombres estaban sacados de las sílabas iniciales de un Himno a San Juan Bautista, llamado «Ut queant laxis»" (Grabner, 2008, p.10).

Sistema Latino	Origen	Sistema inglés	Sistema Latino	Origen	Sistema inglés
Do	Ut quent laxis	C	Sol	Solve Polluti	G
Re	Resonare fibris	D	La	Labii reatum	A
Mi	Mira gestorum	E	Si	Sancte Ioannes	B
Fa	Famuli tuorum	F			

Tabla 3.1 Nomenclatura de la notas musicales.

Los nombres de las notas se tomaron de la primera sílaba de cada verso (ver tabla 3.1), años después Giovanni Battista Doni cambió ut por do.

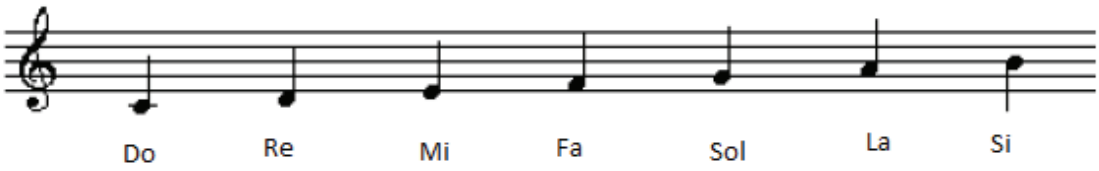


Fig. 3.1 Símbolos de las notas musicales.

La representación de las notas musicales se hace a través de un pentagrama. El pentagrama es un sistema de transcripción musical que consiste en cinco líneas paralelas horizontales. La localización de las notas musicales en el pentagrama viene especificada por la clave utilizada, el nombre de la clave determina cual nota debe situarse en la segunda línea inferior del pentagrama. En la figura 3.1 se puede observar un pentagrama y la ubiacion de las notas musicales en clave de Sol. La figura 3.2

Existen siete claves (do, re, mi, fa, sol, la, si) pero la clave más utilizada es la clave de sol. Su evolución es la siguiente:

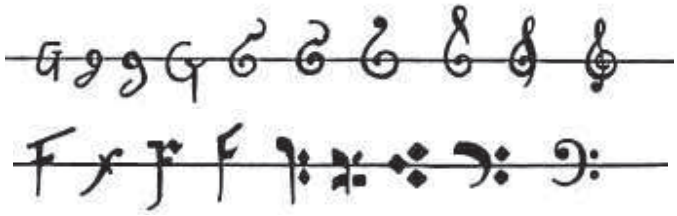


Fig. 3.2 Evolución de las claves de Sol y Fa

"La otra clave más usada, después de la de sol, es la clave de fa o clave de bajo, cuyo signo se formó a partir de la letra F" (Grabner, 2001, p.13).

2. La escala es toda la gama de las frecuencias de las notas percibidas por el oído humano. Desde la nota La más grave hasta el Do más agudo. Para poder precisar la localización que le corresponde a una nota dentro de una escala general, se ha dividido en series de siete notas. A estas series se les llama octavas, iniciando desde la más grave. Estos números que indican las posiciones de la notas se llaman índices acústicas. "El más utilizado es la escala de Franco-Belga que asigna el número -2 a la octava más grave y el 6 a la octava más aguda, la escala se salta el 0 pasando del -1 al 1. La octava a la que pertenece la nota se simboliza incluyendo como subíndice el índice acústico (ejemplo Do_{-2}).

   "El tono es la distancia mayor de entonación que puede haber entre dos sonidos consecutivos no alterados, mientras que el semitono se refiere a la distancia menor de entonación en la misma situación" (Salcedo, 2005, p.65).

   Las alteraciones a los signos que se utilizan para variar la ubicación de las notas. Son cinco, en la tabla 3.2 se describen estas alteraciones:

Sostenido	♯	Sube el sonido un semitono
Bemol	♭	Baja el sonido un semitono
Doble sostenido	𝄪	Sube el sonido dos semitono
Doble bemol	𝄫	Baja el sonido dos semitono
Becuadro	♮	Anula el efecto de las otras alteraciones

Tabla 3.2 Alteraciones en la notas musicales.

Hay dos tipos de alteraciones: las propias y las accidentales. Se llaman propias a las que se colocan al principio del pentagrama y alteran todas las notas de igual nombre en todo el pentagrama; se llaman accidentales a las

que se colocan delante de una nota y alteran las de igual nombre que se encuentran después de ésa en el espacio de un compás.

3. Símbolos son las diferentes figuras que se usan para indicar las diferentes duraciones en las notas. “Existen siete figuras básicas para indicar las diferentes duraciones en las notas” (Salcedo, 2005, p.67). La tabla 3.3 contiene los símbolos para las duraciones por orden decreciente:

Redonda	Blanca	Negra	Corchea	Semicorchea	Fusa	Semifusa
						

Tabla 3.3 Símbolos de las duraciones en la notas musicales.

La relación entre las duraciones es la mitad del símbolo superior, o tomando en cuenta la duración más corta es el doble. Existen también las duraciones en cuanto a silencios, estos corresponden a los mismos tiempos que las duraciones de las notas. Los símbolos para las duraciones de los silencios se muestran en la tabla 3.4 y la figura 3.3 la representación jerárquica de la división de un nota redonda hasta las semifusas.

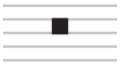
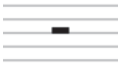
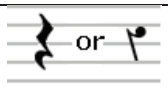
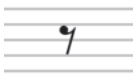
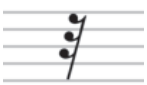
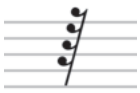
Redonda	Blanca	Negra	Corchea	Semicorchea	Fusa	Semifusa
						

Tabla 3.4 Símbolos de las duraciones de los silencios.

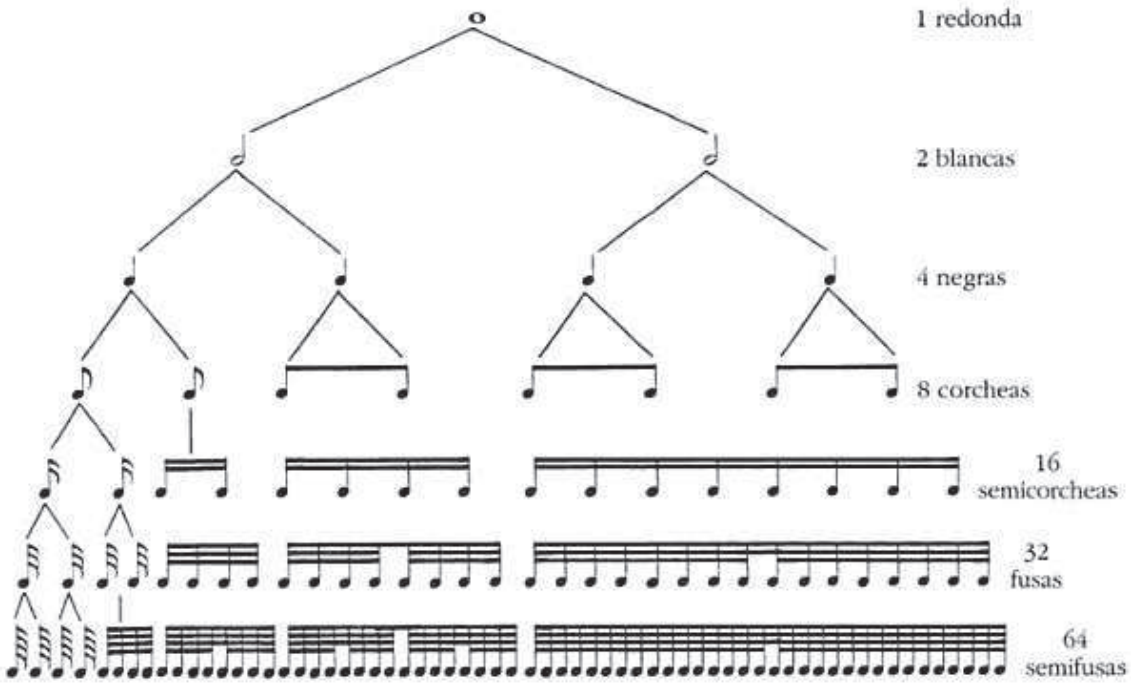


Fig. 3.3 Representación jerárquica de las duraciones en las notas musicales.

"Los términos tradicionalmente utilizados para indicar el movimiento o grado de velocidad con que debe interpretarse una obra musical son palabras italianas que se colocan sobre el pentagrama, al principio de las composiciones" (Grabner, 2001, p.37).

En música se llama "Matices" a un conjunto de signos que se colocan en la partitura con la función de indicar la intensidad relativa de una nota, una frase, o de un pasaje entero. La sucesión de matices constituyen la dinámica de la obra.
"La dinámica gradual estriba en el contraste entre los conceptos de fuerte y suave. Los distintos grados de intensidad pueden recibir a su vez distintas matizaciones" (Grabner, 2001, p.34).

"La dinámica de transición se refiere al aumento o a la reducción paulatina de la intensidad de las notas" (Grabner, 2001, p. 34).

3.3 Tono y semitono

El sistema de afinación temperada divide equitativamente la octava en doce sonidos. La distancia o diferencia en frecuencia sonora entre cada uno de estos sonidos se conoce por el nombre de semitono. En el teclado del piano hay la distancia de un semitono entre teclas contiguas (ver figura 3.4):

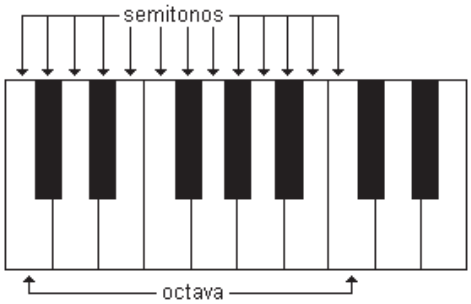


Fig. 3.4 Representación de una octava en el piano.

Un tono equivale a 2 semitonos. Todas las teclas blancas del piano separadas por una tecla negra, están a la distancia de un tono. Las que no tienen tecla negra ente ellas están a un semitono de distancia (ver figura 3.5):

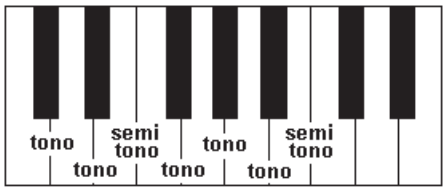


Fig. 3.5 Representación de tonos y semitonos en el piano.

Los sonidos correspondientes a las teclas blancas del piano, reciben los nombres de do, re, mi, fa, sol, la y si. Estas notas se consideran naturales. Podemos alterarlas un semitono cromático ascendente con un sostenido (#) o descendente con un bemol (b) (ver figura 3.6).

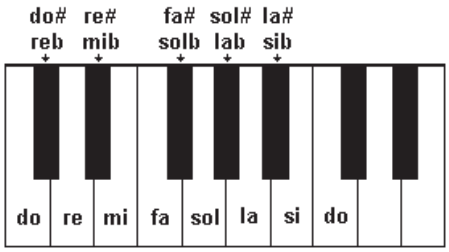


Fig. 3.6 Representación de notas musicales y sostenidas en el piano.

3.4 Afinación

Existen varios métodos matemáticos de cálculo para obtener las frecuencias fundamentales de las notas en relación con las octavas.

3.4.1 Sistema de afinación Pitagórica

Establecida por Pitágoras (580-495 A.C.), la Afinación pitagórica, sistema de construcción de la escala musical que se fundamenta en la quinta perfecta de razón 3/2 o quinta justa; esta afinación era la usada durante la Edad Media.

Esta relación establece que la separación entre un tono corresponde exactamente a 9/8 y para un semitono 256/243. La expresión puede representarse como sigue

$$f(N_{aguda}) = f(N_{grave})\frac{9}{8}\ ; para\ un\ tono$$
$$f(N_{aguda}) = f(N_{grave})\frac{256}{243}\ ; para\ un\ semitono$$

3.4.2 Sistema de afinación temperado.

Este sistema divide una octava en 12 semitonos, conocidos como tonos temperados. La expresión matemática es la siguiente, cosiste en tomar en cuenta la separación en semitonos:

$$f(N_{aguda}) = f(N_{grave})\left(1 + \frac{\#\ semitonos}{12}\right)$$

3.4.3 Sistema de afinación de los físicos

Dado que las proporciones en las relaciones de frecuencia de algunos grados de la escala pitagórica resultaban excesivamente complejas, los físicos las redujeron a otras más sencillas, basándose en la serie de armónicos. Es esta escala la frecuencias de las notas corresponden con los armónicos de las notas de los octavos inferiores. Las frecuencias de las notas de cada octava se pueden calcular a partir del Do de la misma octava.

$$f(N) = \begin{cases} \dfrac{8 + \#semitonos}{8} f(Do_i) & i = 1,2,4 \\ \dfrac{5 + \#semitonos}{6} f(Do_i) & i = 3,5,7 \\ \dfrac{15}{8} f(Do_i) & i = 6 \end{cases}$$

3.5 Ritmo y métrica

"El ritmo es el orden y proporción en el espacio y tiempo" (Torres, 1972, p.15). El ritmo es una estructura de sonidos en el tiempo independientes de la altura y el tono.

En música podemos encontrar cinco tipos distintos de ritmo:

1. Ritmo de valores. Es el que esta determinado por la relación entre la duración de las notas ejecutadas sucesivamente. Está relacionado con los símbolos de silencios y la métrica.
2. Ritmo melódico. Formado principalmente por notas con diferente tono. Las notas más agudas y más graves inducen al oyente una atención especial, recreando un ritmo determinado.
3. Ritmo dinámico. Se produce por los cambios de intensidad, ya sea por segmentos de la pieza completa o por la pieza completa.
4. Ritmo armónico. Se produce cuando existen diferentes voces en una pieza musical. Los ritmos de las voces pueden reforzarse (efecto conclusivo), o por el contrario se anulan (efecto suspensivo).
5. Ritmo constructivo. Se produce por el equilibrio entre los contrastes y coincidencias entre los elementos que componen la obra musical.

El timbre es uno de los parámetros que influyen en el ritmo. "El compás es la menor de las unidades superiores, en las que se agrupan varios tiempos… En la teoría musical (Riemann), la parte que trata del compás y su papel como punto de partida en la articulación de las formas musicales es la métrica" (Grabner, 2001, p.39). La métrica trata la estructura del ritmo en sus diferentes combinaciones, tomando como unidad de tiempo el compás. Cada compás contiene al igual que el ritmo el elemento de la periocidad, lo que implica que también se define la acentuación.

Un elemento muy importante en el ritmo es la fórmula rítmica, esta permite en muchos casos identificar estilos musicales. Las fórmulas rítmicas son porciones de ritmo con entidad y sentido propio, independiente del espacio temporal y del compás que utilice.

3.6 Instrumentos musicales

Se conoce como instrumento musical, en este marco, al objeto dotado con una o más estructuras resonantes y con las características necesarias que le permitan vibrar, de manera tal que pueda producir sonidos en uno o varios tonos.

Existen diferentes clasificaciones de los instrumentos musicales. La clasificación clásica divide a los instrumentos en tres clases diferentes, aunque este sistema es el más aceptado es muy poco preciso:

1. Viento. Los instrumentos de viento generan un sonido cuando se hace vibrar una columna de aire dentro de ellos. La frecuencia de la onda generada está relacionada con la longitud de la columna de aire y la forma del instrumento, mientras que la calidad del tono del sonido generado se ve afectada por la construcción del instrumento y el método de producción del tono.
2. Cuerda. Los instrumentos de cuerda generan un sonido cuando la cuerda es pulsada. La frecuencia de la onda generada depende generalmente de la longitud de la porción que vibra de la cuerda, la tensión de cada cuerda y el punto en el cual la cuerda es tocada; la calidad del tono varia en función de cómo ha sido construida la cavidad de resonancia.
3. Percusión. Los instrumentos de percusión crean sonido con o sin afinación, cuando son golpeados, agitados o frotados. La forma y el material de la parte del instrumento que es golpeada y la forma de la cavidad de resonancia, si la hay, determinan el sonido del instrumento.

En 1914, los musicólogos Erich M. Von Hornbostel y Curt Sachs idearon una clasificación mucho más lógica que pretendía englobar a todos los instrumentos existentes. Esta clasificación es mucho más precisa, ya que tiene en cuenta los principios acústicos que hacen sonar a los diferentes instrumentos.

Así, se establecen cinco grandes clases de instrumentos musicales, que a su vez se dividen en grupos y subgrupos (Von HornBostel y Sachs, 1914), la tabla 3.5 define esta clasificación, y la figura 3.7 el rango de frecuencia para algunos instrumentos.:

Tipo	Definición	Modo de ejecución	Ejemplos
Aerófonos	El sonido se produce al vibrar una columna de aire	Boquilla	Tuba, tromba, trompeta, trombón, corneta
		Bisel	Flauta trasversa,

			piccolo
		Lengüeta simple	Clarinete, saxofón
		Lengüeta doble	Oboe, corno ingles, fagot, contrafagot
		Lengüeta libre	Armónica, acordeón
		Mixta	Órgano de iglesia, gaita
Cordófonos	El sonido se produce al vibrar una cuerda tensa	Frotada	Violín, viola, violoncelo, contrabajo
		Pulsada	Guitarra, laúd, bandurria, balalaika, banjo, ukelele, timple, citara, arpa
		Percutida con teclado	Piano, clavicordio
Idiófonos	El sonido se produce al vibrar el propio cuerpo del instrumento	Entrechoque	Claves, castañuelas, látigo, platillos
		Golpeados	Triángulo, plato, caja china, laminas, campanas, gong, cencerros
		Sacudidos	Sonajero, sistro, maracas, tubos, cascabeles
		Raspados	Güiro, matracas, raspador
		Punteados	Caja de música, arpa de boca
		Frotados	Armónica de cristal, serrucho
		Soplados	Piano chanteur
Membranófonos	El sonido se produce al vibrar una membrana	Percutidos	Timbales, tambor, pandero, bombo,bongós
		Frotados	Tambores de fricción, zambomba
		Soplados	Mirlitón, silbato, kazoo
Electrófonos	El sonido se produce por medios eléctricos	Tradicionales	Piano eléctrico, saxo midi, bajo eléctrico, guitarra eléctrica
		Nuevos	Sintetizador, themerin

Tabla 3.5 Clasificación actual de los instrumentos musicales.

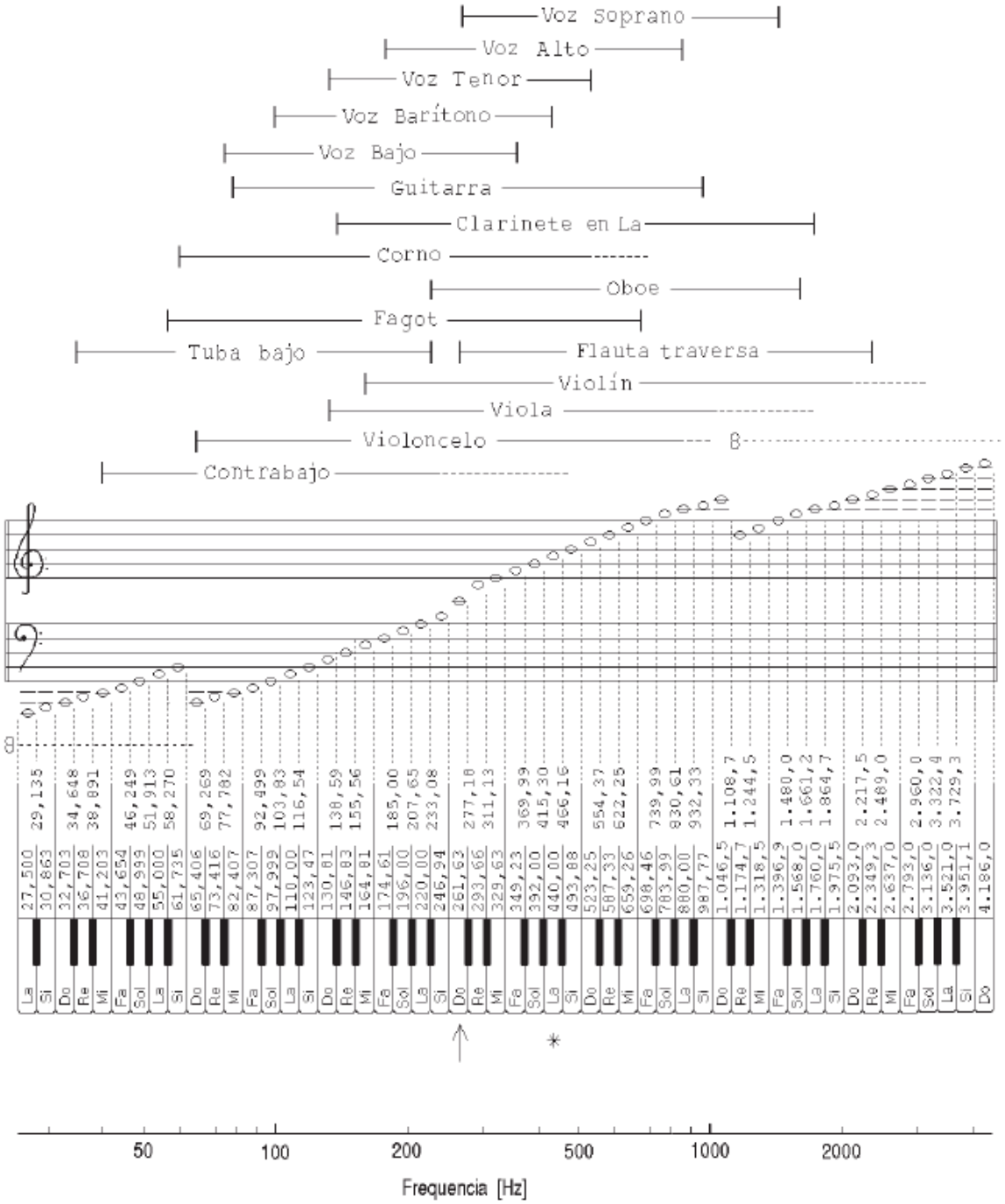


Fig. 3.7 Rango de frecuencias de voces y algunos instrumentos musicales, con referencia en la escala de un piano con sus correspondientes frecuencias.

CAPÍTULO 4

ORGANIZACIÓN Y ADMINISTRACIÓN DE LAS BASES DE DATOS

La Base de Datos es un mecanismo de organización de información muy sofisticado. Está permite almacenar cantidades de datos de importancia en un negocio, una aplicación o una decisión, permite recuperar datos completamente o que cumplan ciertas condiciones, facilita la distribución de los datos balanceadamente para su análisis, estudio o procesamiento.

Los beneficios de las bases son muchos, si se saben aprovechar correctamente, todo inicia desde un minucioso análisis para el diseño de los datos. Saber que datos y que tipos de datos de datos se van almacenar suele tener ventajas en la distribución de la información, evitar sobrecarga de datos, administrar correctamente el espacio de disco y obtener resultados correctos de las consultas.

En este apartado se intenta, además de obtener como resultado una base de datos estable, desarrollar un correcto diseño de la estructura interna, gestionar correctamente los recursos a mi disposición para obtener beneficios extras como mejorar el tiempo de respuesta de consultas, adaptar correctamente los espacios de memoria y crear inserciones perfectamente organizadas.

Para lograr estos objetivos se cuenta con herramientas de gestión de bases de datos, de diseño y los propios lenguajes de manipulación y definición de datos. Si bien la base de datos global no constituye un reto para su creación, por el hecho de manejar pocas tablas, algunas de ellas incluso con muy pocos registros; si se requiere especial cuidado en aquellas cuyos registros son de mucha importancia para la etapa de análisis de datos. Y por tanto el mayor tiempo de procesamiento lo ocuparan estos métodos de análisis y los tiempos de consultas deben ser lo más cortos posibles.

Las tareas que se debe realizar son: selección del sistema manejador de bases de datos, creación de la base de datos, control de acceso a usuarios, creación y asignación de privilegios, carga de datos, creación de procedimientos almacenados, ejecución de programas de inserción de datos, monitoreo de usuarios, rendimientos de consultas, crecimiento de estadísticas y optimización.

4.1 Conceptos teóricos

4.1.1 Sistemas de bases de datos

Date define a los sistemas de bases de datos como “un sistema computarizado para guardar registros” (Date C. J., 2001, p. 27). El sistema cumple diferentes funciones de acuerdo al manejo de los datos almacenados:

- ✓ Agregar nuevos archivos vacíos a la base de datos
- ✓ Insertar datos dentro de los archivos existentes
- ✓ Recuperar datos de los archivos existentes
- ✓ Modificar datos en archivos existentes
- ✓ Eliminar datos en archivos existentes
- ✓ Eliminar archivos existentes de la base de datos

Un sistema de bases de datos se compone de diferentes elementos: datos, hardware, software y usuarios.

Los datos son el conjunto de registros que integran a la base de datos, deben cumplir ciertos criterios de selección para poder ser parte de las base.

El hardware, Date establece los componentes que lo conforman; “volúmenes de almacenamiento… que se emplean para contener los datos almacenados, junto con los dispositivos asociados de E/S. Y los procesos de hardware y la memoria principal asociada para apoyar la ejecución del software del sistema de bases de datos” (Date C. J., 2001, p. 29).

La capa de software conocida como el administrador de base de datos o el servidor de base de datos, es la encargada de procesar todas las solicitudes procedentes de los usuarios. “El DBMS es, por mucho, el componente de software más importante en general, aunque no es el único. Otros comprenden las utilerías, herramientas de desarrollo de aplicaciones, ayudas de diseño, generadores de informes y el administrador de transacciones o monitor PT” (Date C. J., 2001, p.30).

Los usuarios se dividen en tres clases distintas:

1. Programadores, son los responsables de escribir programas de aplicación de bases de datos en algún lenguaje de programación.
2. Usuarios finales, son aquellos que interactúan con el sistema de bases de datos a través de aplicaciones en línea, o bien puede usar una interfaz proporcionada como parte integral del software del sistema de bases de datos.

3. El administrador de base de datos, encargado de gestionar los recursos de la base de datos.

4.1.2 Bases de datos

Date C. J., 2001, “una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación” (p. 32).

Los datos persistentes son aquellos que una vez dentro de la base de datos, solo pueden ser modificados, eliminados por medio de una solicitud explicita de un usuario al DBMS.

Existen elementos que conforman el diseño de una base de datos, la entidad y el vínculo. “La entidad se refiere a cualquier objeto distinguible que va a ser representado en la base de datos… Los vínculos asocian dichas entidades…, son parte de los datos tanto como lo son las entidades” (Date C. J., 2001, p. 34).

4.1.3 Datos y modelos de datos

Citando a Date, 2001:

> “La palabra datos se deriva del vocablo latín para *da*; por lo tanto, los datos son hechos dados, a partir de los cuales es posible inferir hechos adicionales. Un hecho dado corresponde a su vez a lo que en la lógica se denomina proposición verdadera; (p. 25)”

Un modelo de datos es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos nos permiten modelar la estructura de los datos. Los operadores nos permiten modelar su comportamiento.

La independencia de los datos se maneja con tres definiciones pertinentes

“Un campo almacenado es, en general, la unidad más pequeña de datos almacenados. La base de datos contendrá muchas ocurrencias (o ejemplares) de los diversos tipos de campos almacenados” (Date C. J., 2001, p.43).

“Un registro almacenado es un conjunto de campos almacenados relacionados. Una vez más distinguimos entre tipo y ocurrencia. Una ocurrencia (o ejemplar) de registro almacenado consta de un grupo de ocurrencias de campos almacenados relacionados” (Date C. J., 2001, p.44).

“Un archivo almacenado es la colección de todas las ocurrencias existentes actualmente para un tipo de registro almacenado” (Date C. J., 2001, p.44).

4.1.4 Ventajas de las bases de datos

a) Los datos pueden compartirse. Es posible cumplir con los requerimientos de aplicaciones nuevas sin tener que agregar información adicional a la base de datos.
b) Es posible reducir la redundancia. No es necesario crear duplicados de datos para distintas aplicaciones.
c) Es posible evitar inconsistencia de datos. Existe un mecanismo conocido como propagación de actualizaciones que permite reflejar las modificaciones entre aplicaciones de manera automática.
d) Es posible brindar manejo de transacciones. Una transacción es una unidad lógica, que por lo general comprende varias operaciones de la base de datos.
e) Es posible mantener la integridad. Permite asegurar que los datos de la base de datos sean correctos.
f) Es posible hacer cumplir la seguridad.
g) Es posible equilibrar los requerimientos en conflicto.

4.2 Administrador de la base de datos

“El administrador de la base de datos es la persona que proporciona el apoyo técnico necesario para implementar decisiones de estrategia y política con respecto a los datos de la empresa” (Date C. J., 2001, p.63).

Sus funciones son:

a) Definir el esquema conceptual.
b) Definir el esquema interno.
c) Establecer el enlace con usuarios.
d) Definir las restricciones de seguridad y de integridad.
e) Definir las políticas de vaciado y recarga.
f) Supervisar el rendimiento y responder a los requerimientos cambiantes.

4.2.1 El sistema de administración de bases de datos

“Es el software que maneja todo acceso a la base de datos” (Date C. J., 2001, p. 65).

El proceso es el siguiente:

i. Un usuario emite una petición de acceso, utilizando algún sublenguaje de datos específico
ii. El DBMS intercepta esa petición y la analiza
iii. El DBMS inspecciona, en su momento, (las versiones objeto de) el esquema externo para ese usuario, la transformación externa/conceptual correspondiente, el esquema conceptual, la transformación conceptual/interna y la definición de la estructura de almacenamiento.
iv. El DBMS ejecuta las operaciones necesarias sobre la base de datos

Sus funciones concretas son las siguientes:

a) Definición de datos
b) Manipulación de datos
c) Optimización y ejecución
d) Seguridad e integridad de datos
e) Recuperación de datos y concurrencia
f) Diccionario de datos
g) Rendimiento

4.2.2 Utilerías

“Las utilerías son programas diseñados para ayudar al DBA en sus tareas administrativas, algunos programas de utilería operan en el nivel externo del sistema” (Date C. J., 2001, p.72).

Algunos ejemplos de utilería muy usadas:

a) Rutinas de carga, para crear la versión inicial de la base de datos a partir de uno o más archivos del sistema operativo;
b) Rutinas de descarga/recarga, para descargar la base de datos (o partes de ella), para respaldar los datos almacenados y para recargar datos desde dichas copias de respaldo (por supuesto, la "rutina de recarga" es básicamente idéntica a la rutina de carga recién mencionada);
c) Rutinas de reorganización, para reordenar los datos en la base de datos almacenada, por distintas razones que normalmente tienen que ver con el desempeño; por ejemplo, agrupar datos en el disco de alguna forma en particular o recuperar espacio ocupado por datos que se volvieron obsoletos;
d) Rutinas estadísticas, para calcular diversas estadísticas de desempeño, como el tamaño de los archivos, las distribuciones de valores, los contadores de operaciones de E/S, etcétera;
e) Rutinas de análisis, para analizar las estadísticas arriba mencionadas.

4.3 Bases de datos de archivos con notación musical

Las bases de datos para el reconocimiento de notas consisten en un conjunto de archivos de audio, que han sido tratados y ajustados para obtener la mayor cantidad de información relevante.

4.3.1 Notas con duración fija

La base de datos ha sido realizada siguiendo el proceso de la figura 4.1 de tratamiento de datos:

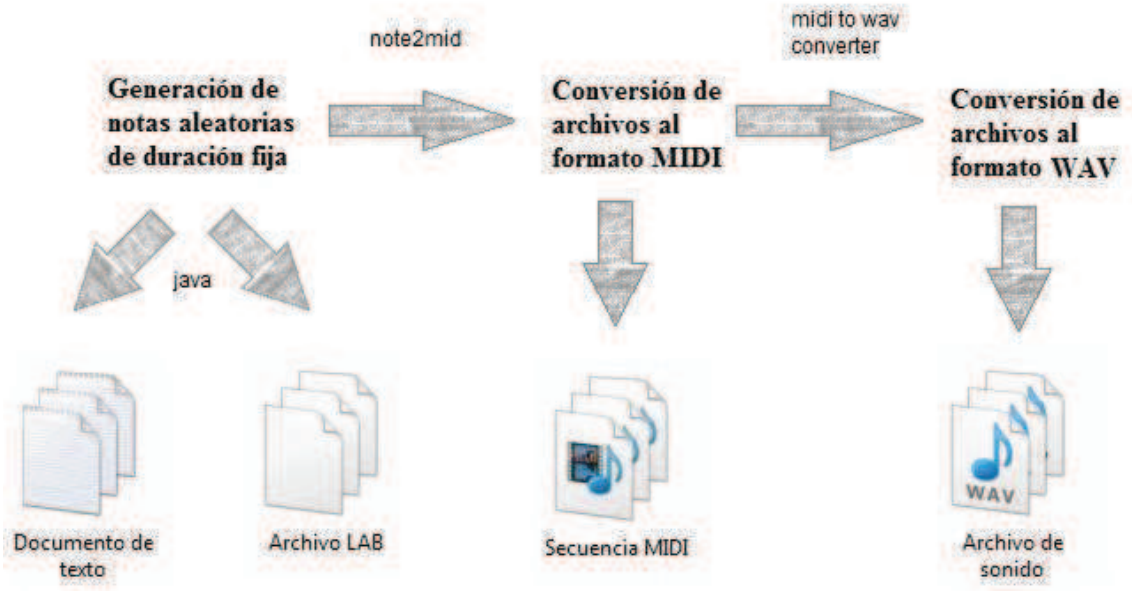


Fig. 4.1 Proceso de creación y transformación de archivos de texto con notas de duración fija a audio digital.

1. La generacion de notas aleatorias fue mediante un programa escrito en Java (la sintáxis del programa completo se encuentra en el Apendice D) y siguiendo la misma estrucutura obtenida a través de la herrmienta *midinote –noheader –values.* Además, aprovecho para generar los archivos de etiquetas de HTK
2. Una vez obtenidos los archivos en texto plano, la herramienta *note2midi* sirve para convertirlos en secuencias MIDI, aplicándolo en un proceso por lotes.

```
@echo off
FOR %%x in (*.txt) DO note2mid %%x
Pause
```

3. Para la conversión de MIDI a WAV, utilice la herramienta *midi to wav converter* que permite la conversión por lotes. Además que también permite convertir a mp3.

Los criterios para la generación de las notas aleatorias es el siguiente:

a) Solo se considera notas semicorcheas, es decir con una duración 1/16.
b) Archivos de 30 segundos cada uno.
c) 100 archivos diferentes.
d) Las notas deben pertenecer a las octavas (2, 3, 4) de la escala Franco Belga
e) No se considera notas alteradas (bemoles y sostenidas).
f) Se consideran 5 instrumentos diferentes para interpretar cada archivo, dando un total de 500 muestras, estos son con su respectiva notación MIDI:
   i. Piano de cola -> piano 1, 0
   ii. Piano brillante -> piano 2, 1
   iii. Guitarra de cuerdas de nylon ->Nylon str. Gt, 24
   iv. Violín -> violin, 40
   v. Flauta -> flute, 73
g) Se consideran silencios.
h) El alfabeto, junto con su código MIDI $\Sigma = \{\ Do2 = 48,\ Re2 = 50,\ Mi2 = 52,\ Fa2 = \ 53,\ Sol2 = 55,\ La2 = ,57\ \ Si2 = 59, Do3 = 60,\ Re3 = 62,\ Mi3 = 64,\ Fa3 = \ 65,\ Sol3 = 67,\ La3 = 69,\ Si3 = 71, Do4 = 72,\ Re4 = 74,\ Mi4 = 76,\ Fa4 = \ 77,\ Sol4 = 79,\ La4 = 81,\ Si4 = 83\}$
i) La distribución de probabilidad por subconjuntos es la siguiente:
   $P\{Do2, Fa2, Do3, Fa3, Do4, Fa4, Si4\} = 0.1795$
   $P\{Siliencios\} = 0.1025$
   $P\{Re2, Mi2, Sol2, La2, \dots\} = 0.718$
j) El programa tiene líneas adicionales para contabilizar la frecuencia de aparición de cada nota de manera global.

   Tras ejecutar una vez el programa se obtuvieron el conteo de la aparición de las notas y el silencio. La grafica de la figura 4.2 representa espectro de aparición de las notas en los archivos de texto y midi.

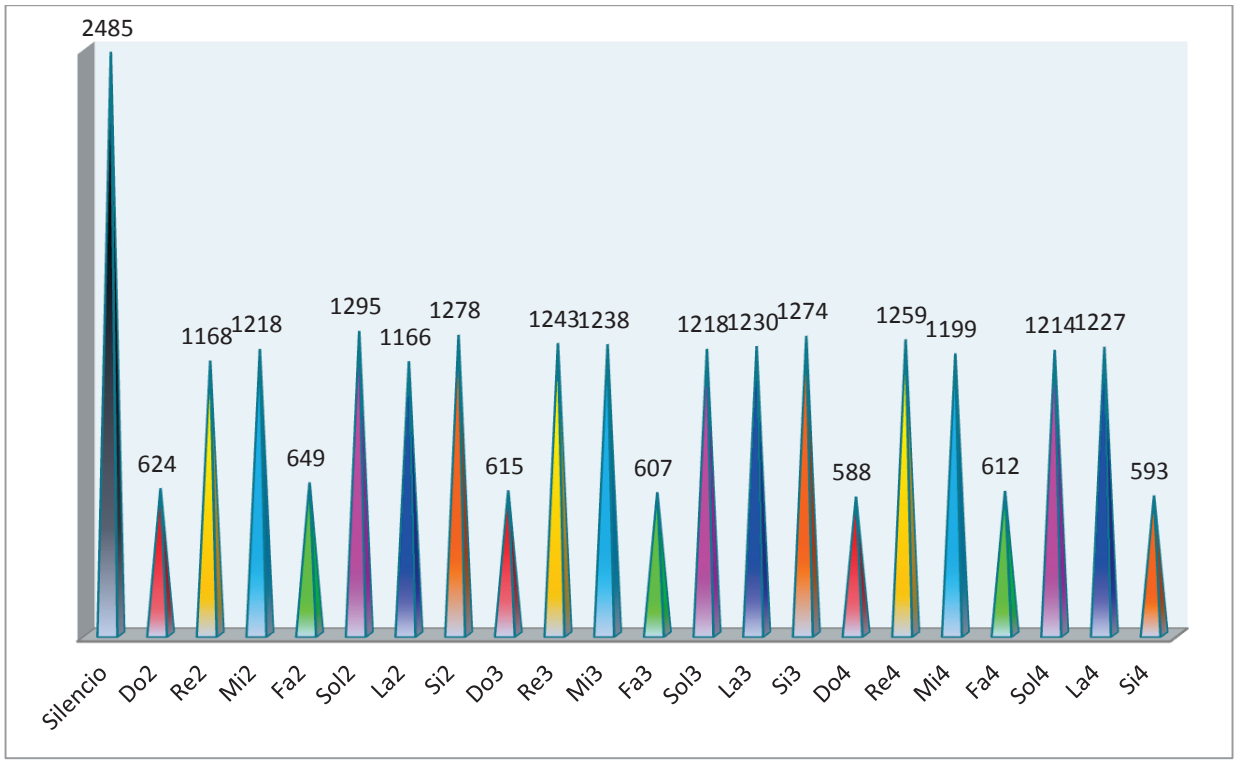


Fig. 4.2 Frecuencia de aparición de las notas musicales con duración fija para un solo instrumento.

4.3.2 Notas con duración variable

El proceso de la figura 4.3 representa la generación de archivos de texto y etiquetas de muestras hasta la conversión a archivos de audio WAV.

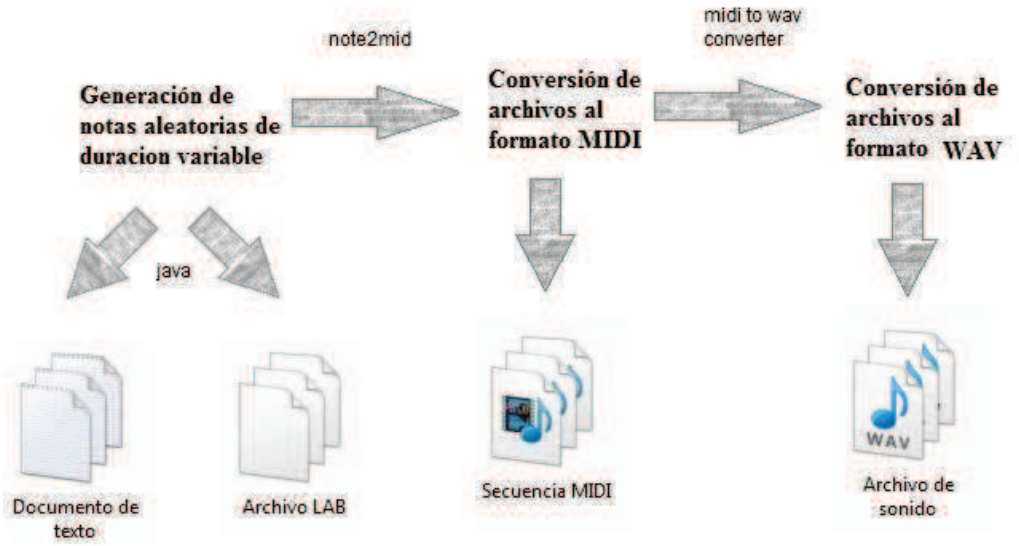


Fig. 4.3 Proceso de creación y transformación de archivos de texto con notas de duración variable a audio digital.

Los criterios para la generación de notas aleatorias con duración variable son los mismos y se añaden los siguientes:

a) Se consideran semicorcheas (1/16), corcheas (1/8), negras (1/4), blancas (1/2) y redondas (1).
b) La distribución de probabilidad para cada figura es la siguiente:
   i. $P(Semicorcheas) = 0.1$
   ii. $P(Corcheas) = 0.18$
   iii. $P(Negras) = 0.24$
   iv. $P(Blancas) = 0.29$
   v. $P(Redondas) = 0.19$
c) El programa tiene líneas adicionales para contabilizar la frecuencia de aparición de cada figura de manera global.

Los datos obtenidos tras la ejecución del programa han sido representados en la graficas de las figuras 4.4 y 4.3, donde se representa frecuencia para las notas musicales y las duraciones, respectivamente.

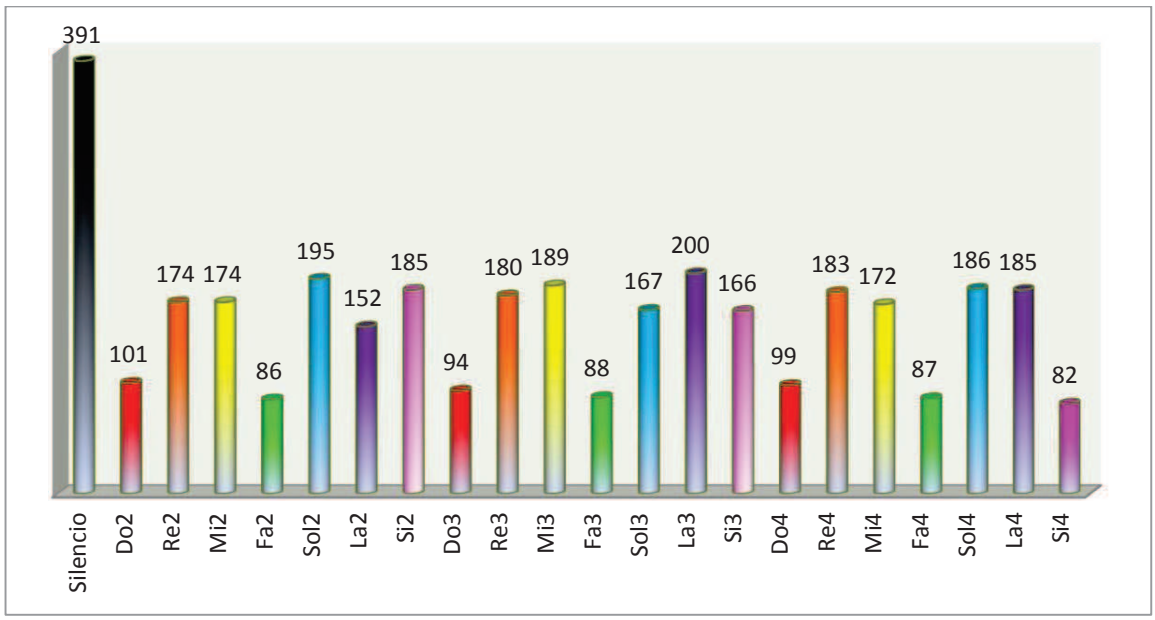


Fig. 4.4 Frecuencia de aparición de las notas musicales con duración variable para un solo instrumento.

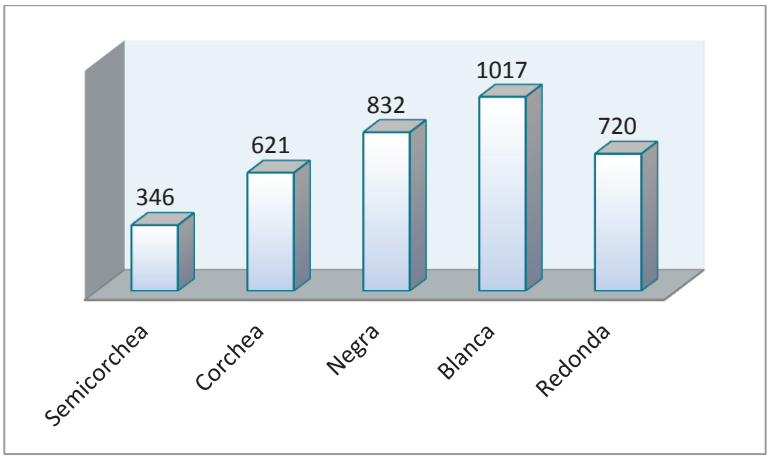


Fig. 4.5 Frecuencia de aparición en los archivos de las diferentes duraciones.

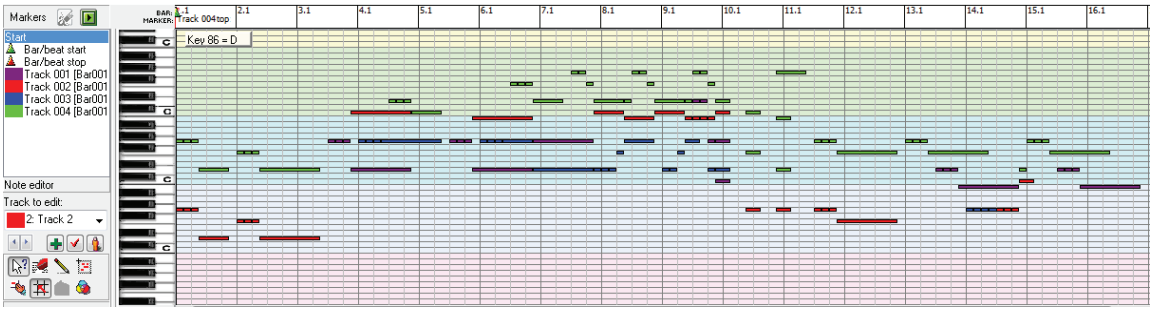
El programa completo que genera los archivos de texto con la estructura de Gauntler se encuentra en el apéndice E.

4.3.3 Música monofónica

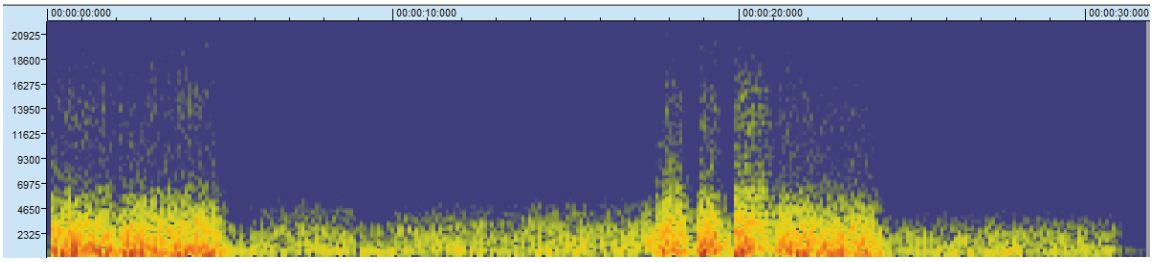
El análisis musical debe hacerse utilizando música real. Los archivos con notas aleatorias y de duración fija y variable, son utilizados como datos de entrenamiento. Pero para una aplicación real es necesario utilizar música real. Sin embargo la mayoría de las piezas musicales, por no decir todas, tienen un amplio rango de notas y emplean muchas octavas. Para fines de este trabajo, los archivos con música real, antes de pasar a música más compleja, deben ser piezas monofónicas.

La recopilación de canciones consta de 10 piezas musicales de distinto género en formato MIDI. A continuación se encuentra el listado de las mismas, descripción de ellas. La figuras de la 4.6 a la 4.15 son representaciones de la estructura MIDI y el espectro de cada una de estas piezas musicales.

EE01. Allegro con brio. Ludwig Van Beethoven. 1804 – 1809. Sinfonía. Instrumentos: Piano de cola, timbales, cuerdas 1, cuerdas 2.



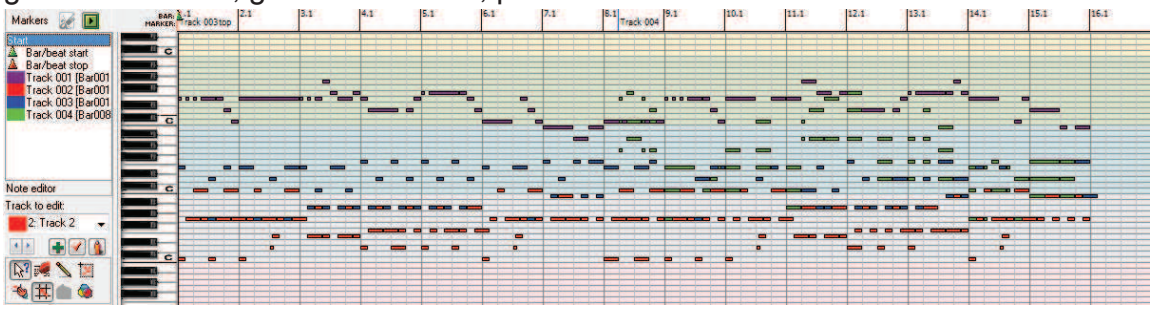
a)



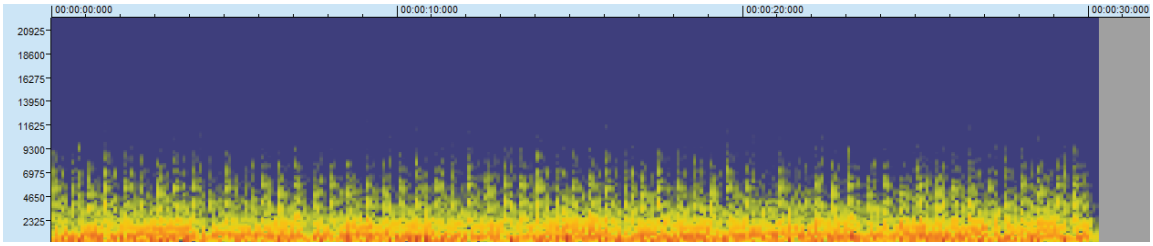
b)

Fig. 4.6 a) Visualización gráfica del MIDI de Allegro con Brio usando 4 instrumentos y b) es su espectrograma.

EE02. Candy Candy. Takeo Watanabe. 1975. Instrumentos: Flauta travesera, guitarra acústica, guitarra eléctrica, piano de cola.



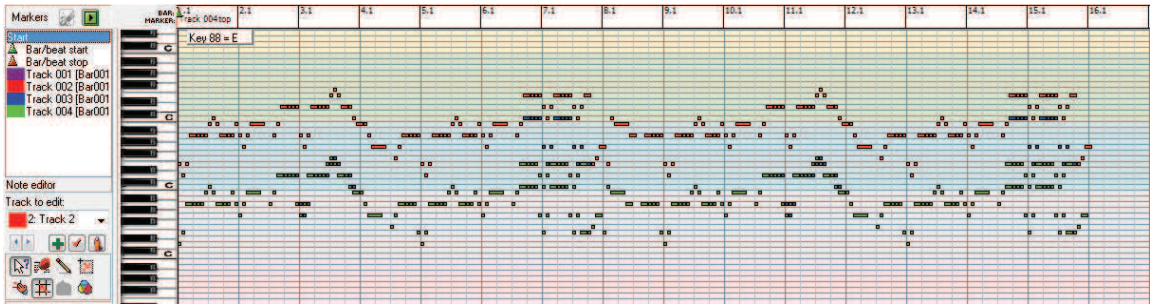
a)



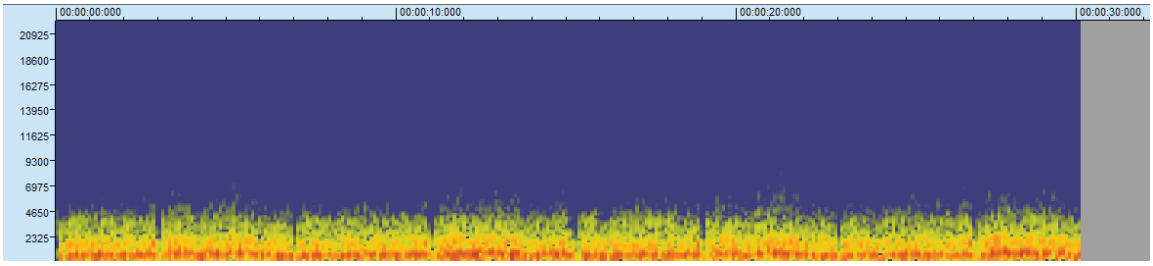
b)

Fig. 4.7 a) Visualización gráfica del MIDI de Candy Candy usando 4 instrumentos y b) es su espectrograma.

EE03. Carmen. G. Bizet. 1845. Ópera. Instrumentos: Flauta travesera, flautín, oboe, clarinete.



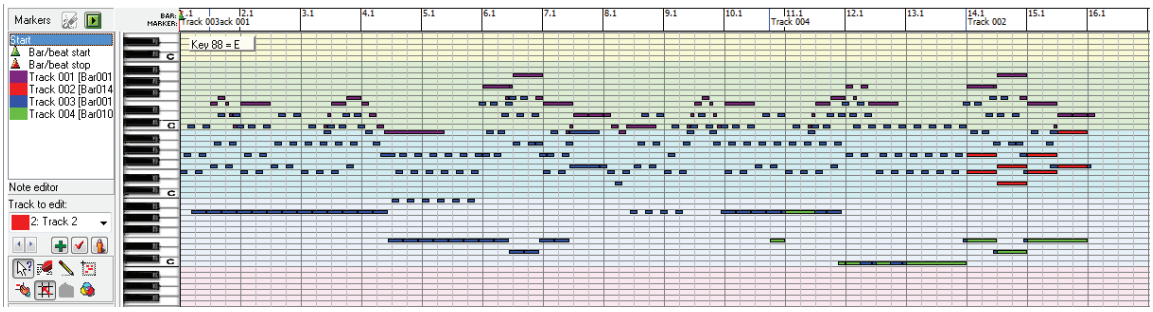
a)



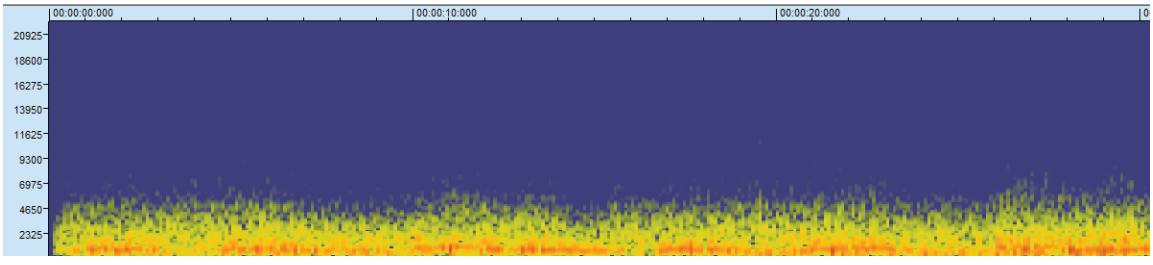
b)

Fig. 4.8 a) Visualización gráfica del MIDI de Carmen usando 4 instrumentos y b) es su espectrograma.

EE04. Chiquitita. Abba. 1979. Pop europeo. Instrumentos: Guitarra eléctrica, cuerdas 1, cuerdas 2, bajo.



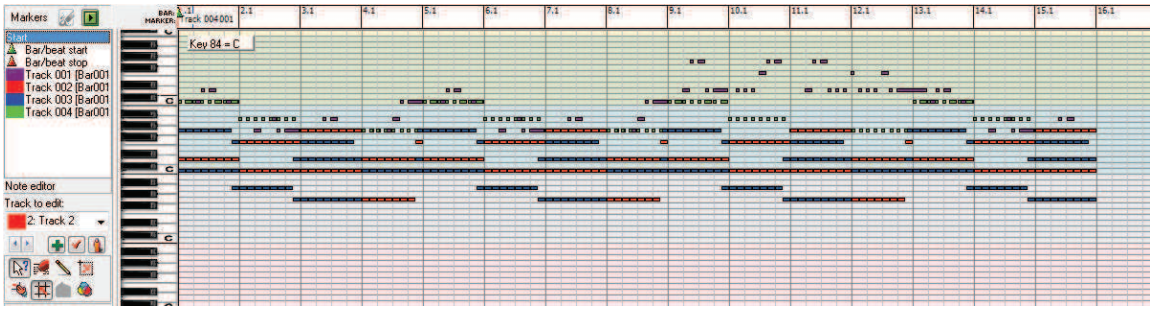
a)



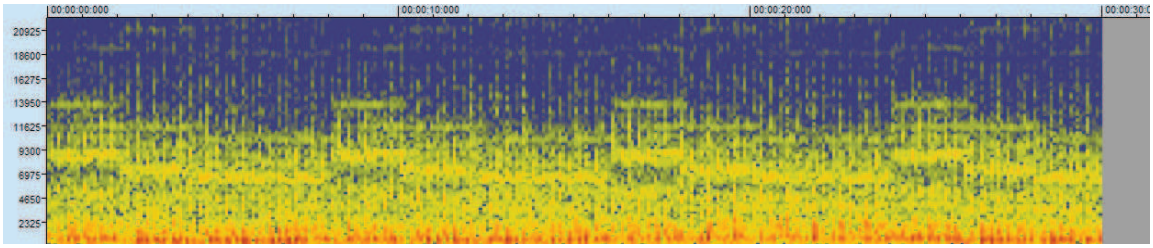
b)

Fig. 4.9 a) Visualización gráfica del MIDI de Chiquitita usando 4 instrumentos y b) es su espectrograma.

EE05. Firework. Katty Perry. 2010. Dance pop. Instrumentos: órgano, de vapor, banjo, clave, citara.



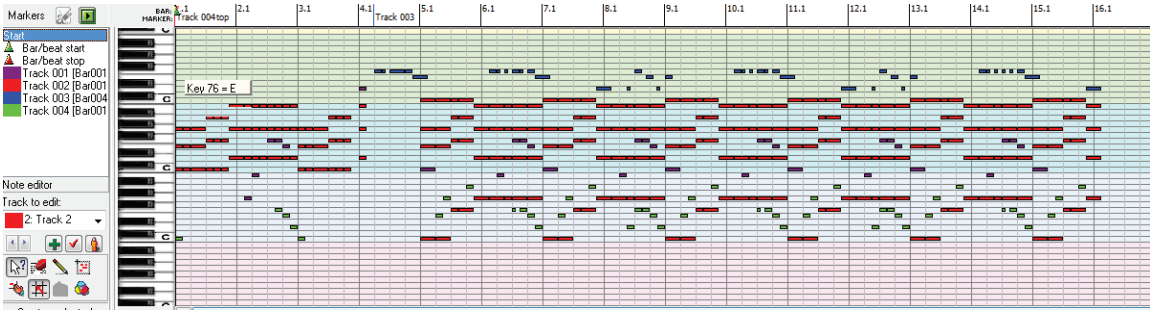
a)



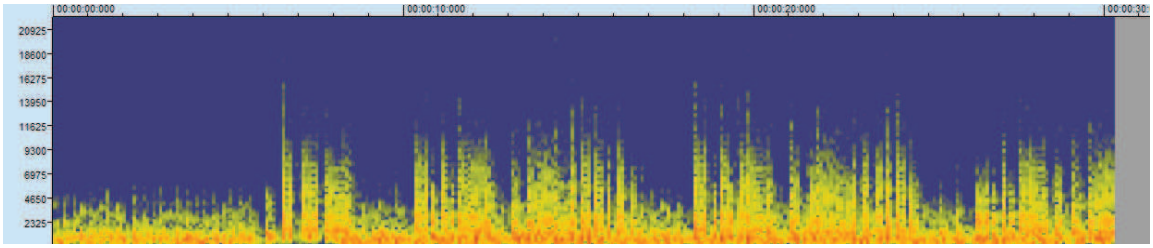
b)

Fig. 4.10 a) Visualización gráfica del MIDI de Firework usando 4 instrumentos y b) es su espectrograma.

EE06. La bamba. Anónimo. Década 50's. Huapango. Instrumentos: Guitarra eléctrica, guitarra española, saxofón contralto, bajo.



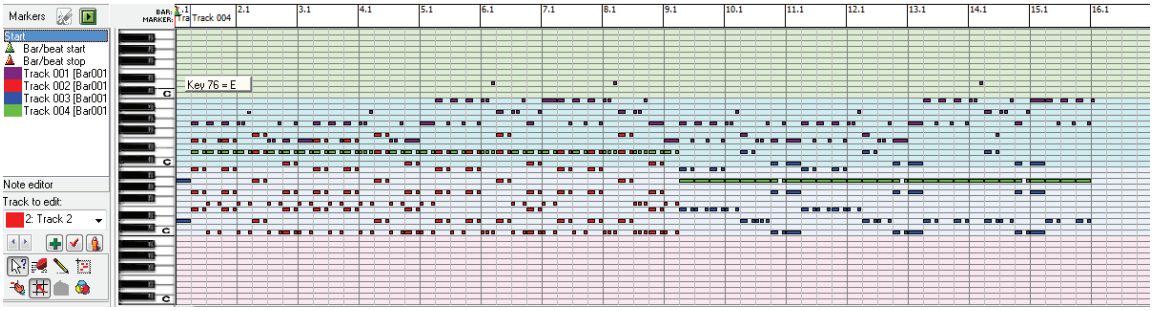
a)



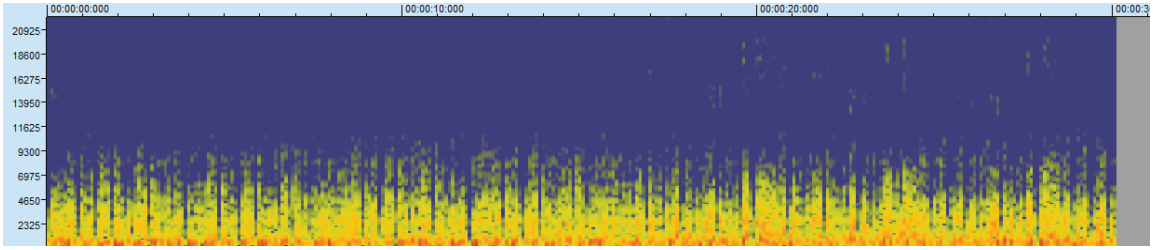
b)

Fig. 4.11 a) Visualización gráfica del MIDI de La Bamba usando 4 instrumentos y b) es su espectrograma.

EE07. Molinos de viento. Mago de Oz. 2002. Folk Metal. Instrumentos: Violín, guitarra acústica, guitarra con distorsión, guitarra española.



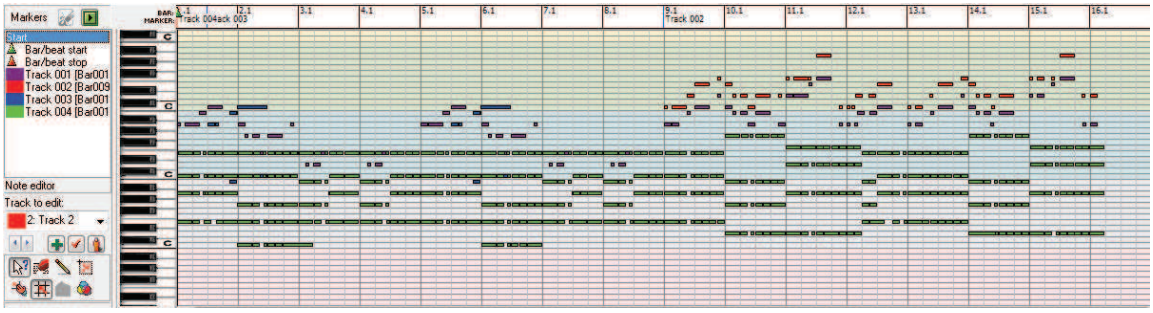
a)



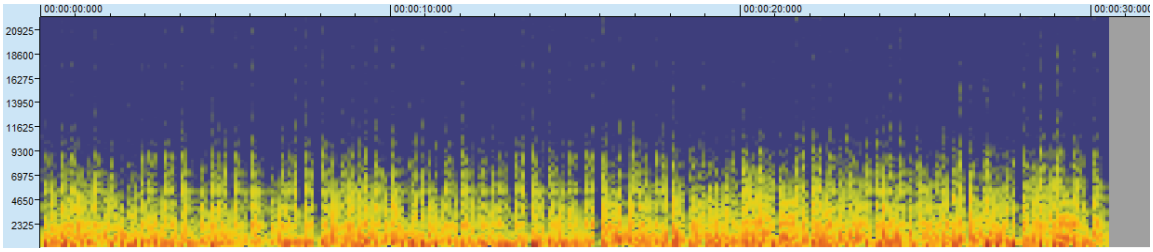
b)

Fig. 4.12 a) Visualización gráfica del MIDI de Molinos de Viento usando 4 instrumentos y b) es su espectrograma.

EE08. No milk today. Herman Hermit's. 1966. Pop. Instrumentos: Saxofón contralto, saxofón soprano, coro aahs, guitarra acústica.



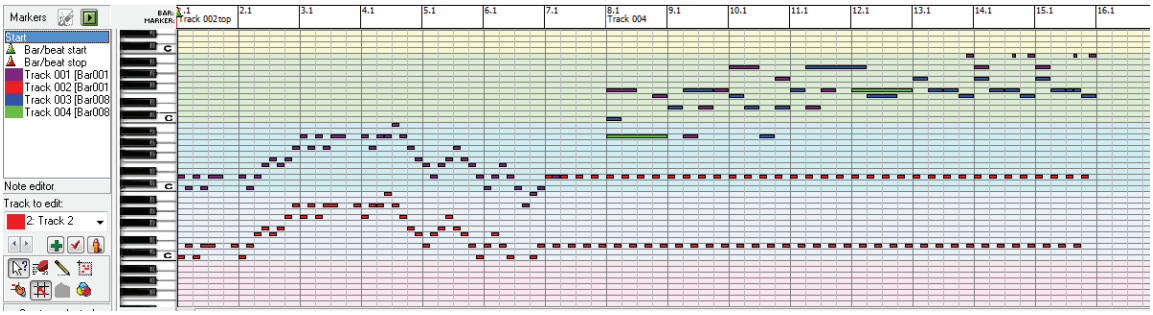
a)



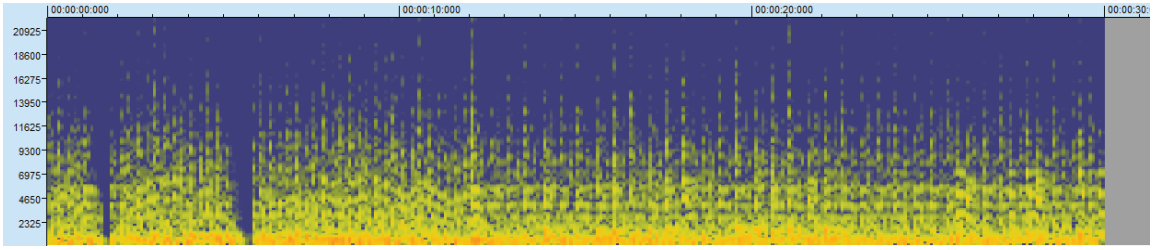
b)

Fig. 4.13 a) Visualización gráfica del MIDI de No milk today usando 4 instrumentos y b) es su espectrograma.

EE09. The Marriage of Figare. Wolfgang A. Mozart. 1786. Ópera bufa. Instrumentos: Piano eléctrico 2, piano eléctrico 1, piano de cola, paino brillante.



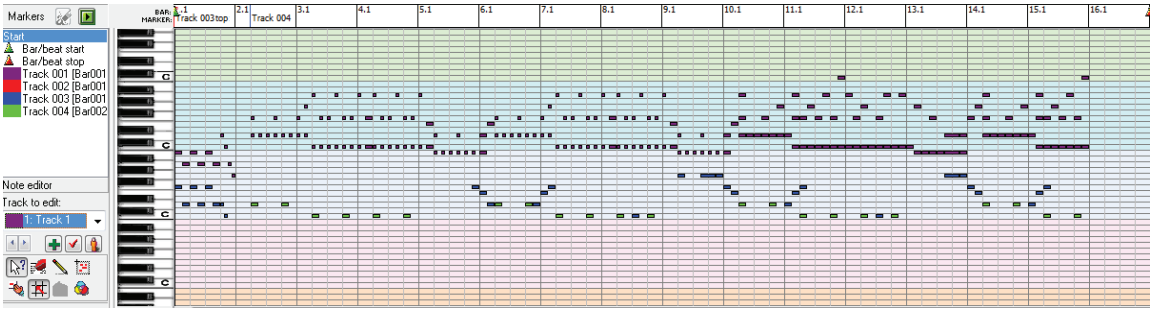
a)



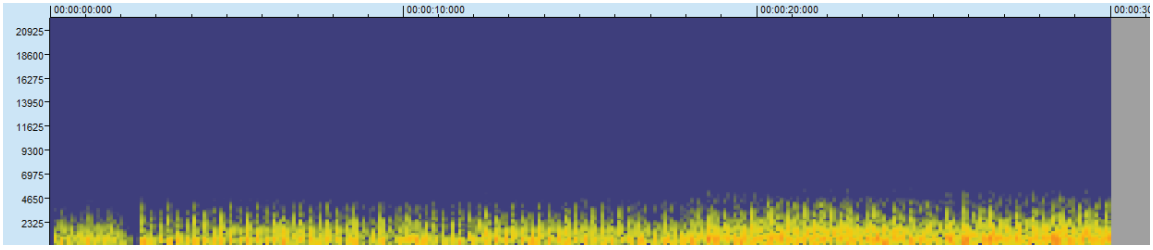
b)

Fig. 4.14 a) Visualización gráfica del MIDI de The marriage of Figare usando 4 instrumentos y b) es su espectrograma.

EE10. Zoosters Breakout. Hanz Zimmer.2005. Sountrack. Instrumentos: Piano de cola, cuerdas 1, piano brillante, bajo acústico.



a)



b)

Fig. 4.15 a) Visualización gráfica del MIDI de Zoosters breakout usando 4 instrumentos y b) es su espectrograma.

Sin embargo para poder usar estas canciones en el proceso, ha sido necesario manipularlos para que cumplan las características favorables para su extracción de parámetros. El proceso de adecuación es el siguiente:

i. Extracción de 30 segundos del archivo
ii. Extracción de la pista con el ritmo destacado de cada archivo
iii. La modificación se manifiesta más en las notas
   a. Uso de las mismas octavas que en los archivos de notas aleatorias (escala franco-belga, octavas 2, 3, 4). Si una nota se encuentra fuera de una de las octavas utilizadas se cambia a la octava más cercana. La frecuencia de aparición para cada nota y el silencio se ve encuntra en la gráfica 4.16.

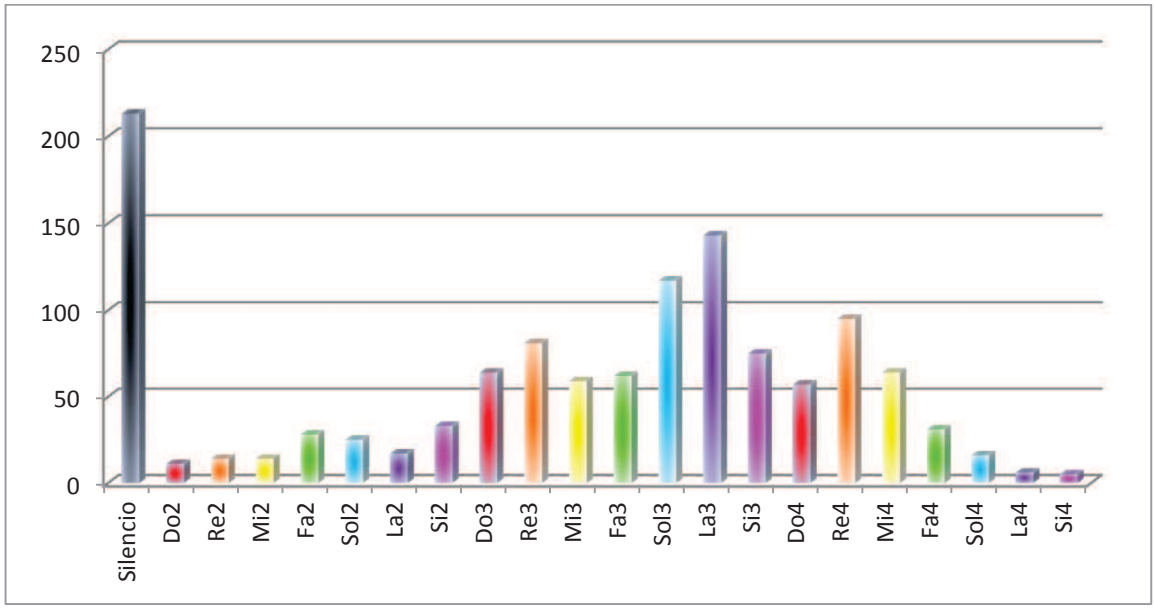


Fig. 4.16 Frecuencia para las notas musicales en archivos de música monofónica.

   b. Para la duración, ha sido necesario normalizar los tiempos de duración, usando los siguientes: semicorchea=125 [ms], corchea= 250 [ms], negra= 500 [ms], blanca= 1000[ms], redonda= 2000[ms]. Si alguna nota tiene una duración diferente se cambia a la más cercana.

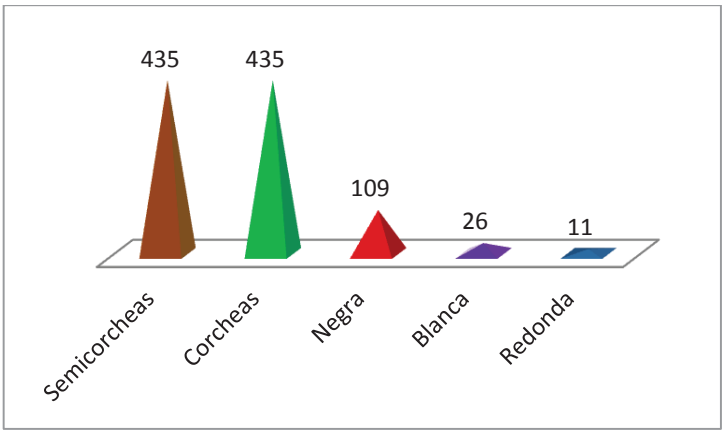


Fig. 4.17 Frecuencia de aparición las diferentes duraciones en archivos de música monofónica.

c. No se consideran fusas ni semifusas, ambas se cambian a semicorcheas. Su distribución puede observarse en la gráfica 4.17.
d. Eliminación de bemoles y sostenidos cambiándolos a su nota natural.

iv. Para cada archivo monofónico resultante, lo grabo con los instrumentos utilizados anteriormente añadiendo 5 más, obteniendo un total de 100 archivos para el reconocimiento. El objetivo de dar variabilidad en los instrumentos musicales, es para crear independencia en cuanto al instrumento.

a. Acoustic grand -> piano 1, 0
b. Bright acoustic -> piano 2, 1
c. Nylon string guitar ->Nylon str. Gt, 24
d. Steel String Guitar -> Steel-str. Gt, 25
e. Violin -> violin, 40
f. Trumpet -> Trumpet, 56
g. Alto sax -> Alto sax, 65
h. Oboe -> Oboe, 68
i. Clarinet, -> Calrinet, 71
j. Flute -> flute, 73

4.3.4 Música polifónica

Para la última etapa, es necesario probar el sistema con música polifónica.

Este tipo de música es un poco más compleja que la anterior, sólo por el hecho de que utiliza un canal diferente para instrumento incluido en la pieza musical.

La preparación de los archivos es similar al proceso utilizado en la música monofónica, sin embargo para la extracción de las diferentes pistas, ha sido necesario normalizar el canal de cada instrumento, limitándolos a 1, 2, 3 y 4.

i. Las piezas musicales son las mismas que las monofónicas.
ii. Extracción de 30 segundos de cada pieza.
iii. Elección de 2, 3 y 4 pistas para cada pieza musical.
iv. Normalización de los canales, a 1, 2, 3 y 4.
v. Normalización de la figura de las notas, siendo: semicorcheas= 125[ms], corcheas= 250[ms], negras= 500[ms], blancas= 1000[ms], redondas= 2000[ms].

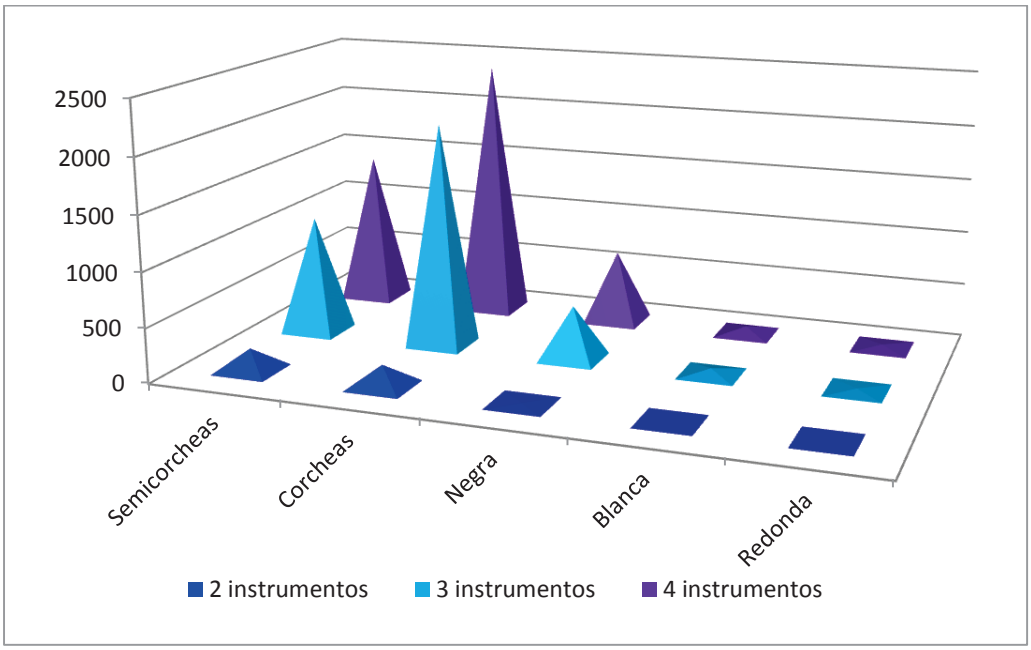


Fig. 4.18 Frecuencia de aparición las diferentes duraciones en archivos de música polifónica para 2, 3 y 4 instrumentos.

vi. Extensión de fusas y semifusas a semicorcheas.
vii. Filtración de las notas en las octavas 2, 3 y 4 de la escala Franco-Belga.

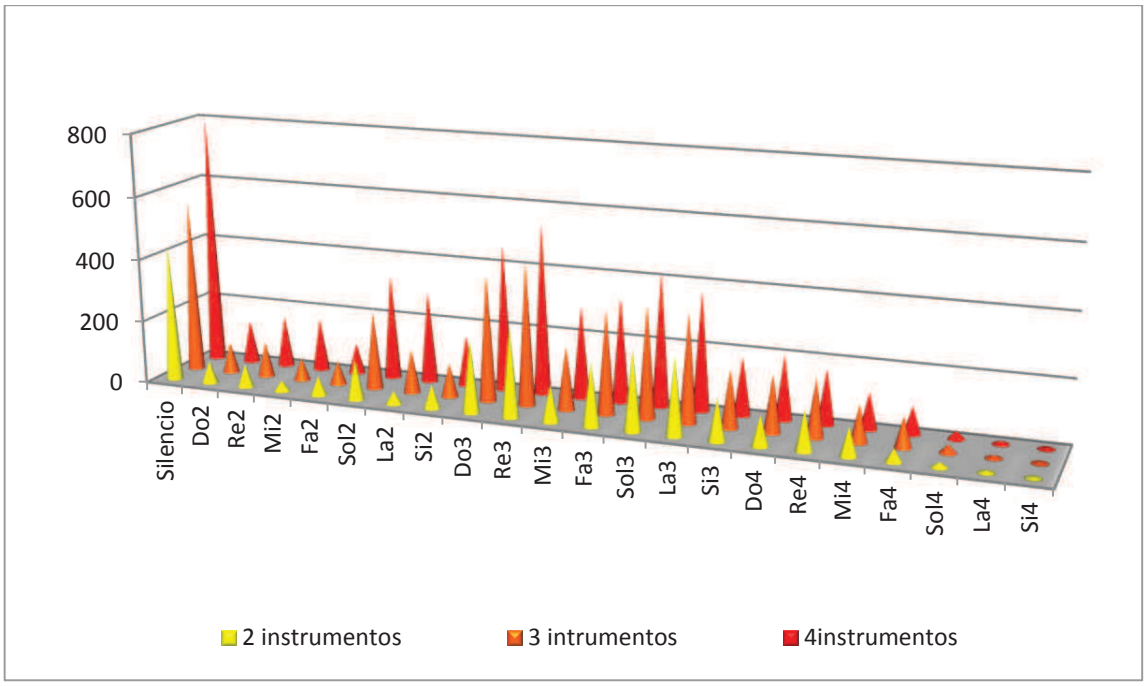


Fig. 4.19 Frecuencia de aparición de notas musicales en archivos de música polifónica para 2, 3 y 4 instrumentos.

Podemos ver en la graficas 4.18 y 4.19 que la tercera octava destaca con mayor número de notas, sin embargo, por tratarse de música la variación de las notas es mucho mayor, que la sugerida en las probabilidades utilizadas para la generación de notas aleatorias. Analizando la tercera octava de manera aislada, podemos observar que guarda cierta relación con las probabilidades propuestas.

Observando las gráficas de las figuras de las notas, bastante erradas las probabilidades propuestas, teniendo una mayor presencia las corcheas que las demás, sin embargo estos errores deben irse puliendo en la etapa de entrenamiento.

CAPÍTULO 5

MODELOS OCULTOS DE MARKOV

Un Modelo Oculto de Markov es un proceso estocástico formado por dos procesos dependientes, el primero es un proceso no observable *q*, de ahí el nombre de modelo oculto, y el segundo es un proceso observable *O* cuyos estados dependen estocásticamente de los estados ocultos de *q*.

Antes de comenzar a explicar el procedimiento conviene especificar algunos conceptos generales de los modelos. Una cadena de Markov $q = \{q_t\}_{t \in N}$ es un proceso estocástico discreto de Markov. La condición fundamental para determinar una cadena de Markov establece que las probabilidades de transición entre estados solo dependan del estado actual y del estado consecuente, y no tomando en cuenta los estados anteriores.

$$P[q_t | q_{t-1}]$$

Matemáticamente hablando la cadena de Markov se define como (Q, A), donde $Q = \{1,2, \dots, N\}$ son los posibles estados en la trayectoria de la cadena, y $A = (a_{ij})$ es una matriz de transición de estados en el modelo. Y la probabilidad de transición se obtiene $a_{ij}(t) = P[q_t = j | q_{t-1} = i]$.

5.1 Definición de los modelos ocultos de Markov

Rabiner, en 1989 en la postulación de su trabajo explicó que un modelo oculto de Markov es:

"… un doble proceso estocástico con un proceso subyacente que no es observable (oculto) pero que puede ser observado a través de otro conjunto de procesos estocásticos que generan la secuencia de observaciones (p.257)".

Un modelo Oculto de Markov es una cadena de estados q junto con un conjunto de valores obtenidos probabilísticamente de un alfabeto $\sum$. El sistema funciona atravesando estados aleatoriamente mientras por cada estado selecciona un símbolos al azar del alfabeto. Cuando se encuentra en el estado $q_{t-1} = i$ tiene la probabilidad de a_{ij} de moverse al estado $q_t = j$ y la probabilidad $bj(k)$ de emitir el símbolo $o_t = v_k$ en el tiempo t.

Únicamente la secuencia de símbolos emitidos en los estados q es observable dejando la trayectoria de estados q oculta.

"Una cadena de Markov $q = \{q_t\}_{t \in N}$ es un proceso estocástico de Markov discreto" (Rabiner, 198, p. 258). Formalmente, una cadena de Markov se define como (Q, A), donde $Q = \{1,2,..N\}$ son los posibles estados de la cadena y $A = (a_{ij})_{nxm}$ es una matriz de transición de estados en el modelo. La condición fundamental de que sea una cadena de Markov es que las probabilidades de transición y emisión dependan solamente del estado actual y no del estado pasado.

Los Modelos ocultos de Markov poseen las siguientes características (Navarrete, 2007, p.38):

a) Los Modelos Ocultos de Markov siempre están en algún estado.
b) El observador de un modelo de Markov, sólo ve el resultado de las funciones aleatorias.
c) Los modelos ocultos de Markov cambian de estado de acuerdo a una matriz de transiciones.

5.2 Elementos de los modelos ocultos de Markov

"Un Modelo Oculto de Markov discreto de primer orden se define con una quíntupla de elementos" (Rabiner, 1989, p.260).

$$\lambda = (\sum, Q, A, B, \pi)$$

i. $\sum = \{v_1, v_2, .., v_M\}$ es un alfabeto o conjunto discreto finito de M símbolos.
ii. $Q = \{1,2,..N\}$ es un conjunto de N estados. Todos los estados deben estar conectados entre sí, de forma que todos tengan la probabilidad de ser alcanzados alguna vez.
iii. $A = (a_{ij})$ Es una matriz cuadrada de dimensión N que contiene probabilidades de transición donde a_{ij} es la probabilidad de transición del estado i al estado j para todo $i, j \in N$.
iv. $B = (b_j(v_t))_{NXM}$ es un vector de probabilidad para cada estado en cada instante de tiempo t, donde $b_j = \{b_{j1}, b_{j2}, \dots, b_{jM}\}$ es la probabilidad de obtener un símbolo del alfabeto en el estado j.
v. $\pi = \{\pi_1, \pi_2, .., \pi_N\}$ es la probabilidad del estado inicial q_0 en Q.

Las probabilidades de iniciación, transición y emisión son los parámetros del modelo. "Dados valores apropiados para N, M, A, B y _, el MOM puede ser utilizado como el generador de una secuencia de observaciones" (Navarrete, 2007, p.42) $O = \{o_1, o_2, .., o_T\}$ donde o_t es uno de los símbolos del alfabeto, T es la longitud de la secuencia de observación y $\lambda = (A, B, \pi)$ son los parámetros del modelo.

El algoritmo es el siguiente:

1. Elegir el estado inicial q_0= S de acuerdo a π
2. $t = 1$
3. Seleccionar $o_t = v_k$ de acuerdo a la distribución de probabilidad de $b_i(k)$
4. Hacer una transición $q_{t+1} = S$ de acuerdo a la distribución de probabilidad a_{ij}
5. $t = t + 1$, si $t < T$ regresa al paso 3
6. Fin

5.3 Topologías

Un Modelo Oculto de Markov puede ser representado como un grafo dirigido con la estructura transición/emisión o un autómata de Melly. La topología más adecuada depende principalmente de la aplicación y permitan modelar de la mejor forma las propiedades o características del sistema.

La clasificación de las arquitecturas depende de las transiciones y conexiones entre los diferentes estados.

Existen tres principalmente,

a) Ergódicas, donde cualquier estado tiene conexión con los demás estados. Figura 5.1. “La matriz de probabilidad de transición de estados es completa” (Navarrete, 2007, p.43).

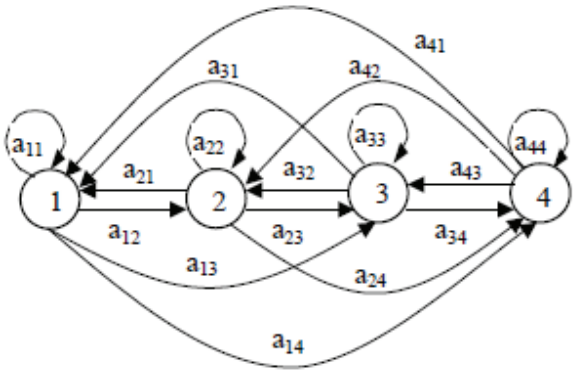


Fig. 5.1 Ejemplo de topología ergódica.

b) Izquierda a derecha, o Bakis, solo permite transiciones a los estados posteriores sin permitir que un estado regrese al estado anterior. Poseen una matriz de transición superior (ver figura 5.2).

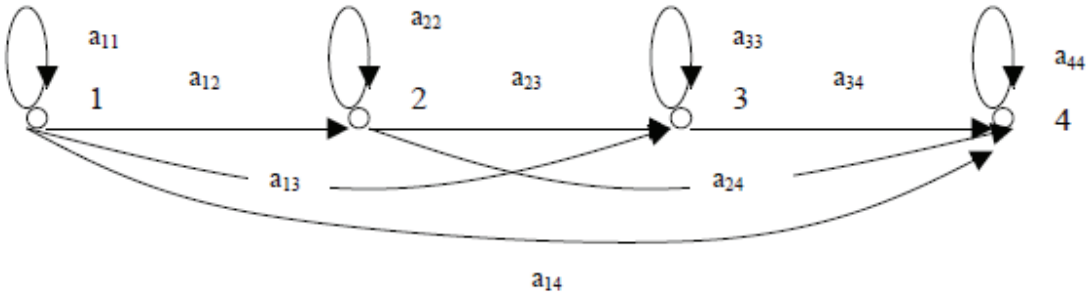


Fig. 5.2 Ejemplo de topología Bakis.

c) Izquierda a derecha paralela, es una arquitecturas izquierda-derecha conectas.

Los Modelos Ocultos de Markov también se pueden clasificar de acuerdo al resultado de los símbolos obtenidos en el proceso, puede ser discretos (Modelos Ocultos de Markov discretos), pueden ser continuos (Modelos Ocultos de Markov Continuos), o semi continuos.

5.4 Problemas de los modelos ocultos de Markov

Rabiner, (1989) estableció que dentro de utilización de los modelos Ocultos de Markov dentro de un sistema de reconocimiento, requiere la solución de tres problemas (p.261).

- Iniciación. Dada una secuencia de observaciones O y un modelo λ, como elegir los parámetros del modelo $\lambda = (A, B, \pi)$ para que la probabilidad de generación de dicha secuencia por el modelo sea óptima.
- Evaluación. Dada una secuencia de observaciones O y un modelo λ, se busca evaluar la probabilidad $P[O|\lambda]$ de que la secuencia haya sido producida por dicho modelo.
- Decodificación. Dada una secuencia de observaciones O como obtener la secuencia de estados S que mejor obtenga la secuencia por parte del modelo λ.

Estos problemas se centran en la fase de entrenamiento y de reconocimiento. En la fase de entrenamiento, a un modelo se le proporciona una secuencia de observaciones de entrenamiento para estimar la matriz de probabilidades de transición A, las distribuciones de probabilidad B y las distribuciones iniciales π.

La solución la evaluación permite evaluar la probabilidad de la generación de la secuencia. Esta probabilidad sirve para clasificar las secuencias de observaciones, lo que constituye la base del reconocimiento.

La decodificación permitirá extraer información sobre el proceso oculto, al obtener la secuencia óptima de estados. Esta información es útil para interpretar el significado de los estados del modelo, así como extraer parámetros estadísticos sobre dichos estados.

5.4.1 Algoritmo Forward-Backward para evaluación

Los Modelo Ocultos de Markov se pueden utilizar para evaluar la probabilidad de que una secuencia de observación O, dado el modelo λ, es decir $P[O|\lambda]$.

“La forma más simple en resolver este problema consiste en determinar todas las posibles trayectorias de los estados y calcular la probabilidad para cada una de las secuencias hasta obtener la probabilidad óptima” (Navarrete, 2007, p.48).

La probabilidad de la secuencia O dada q:

$$P(O|q, \lambda) = \prod_{t=1}^{T} P(O_t|q_1, \lambda)$$

La probabilidad de la secuencia de estados Oes:

$$P(q|\lambda) = \pi_{q1} a_{q1q2} \dots a_{qT-1qT}$$

La probabilidad de O sobre todas las posibles secuencias $q = (q_1, q_2, \dots, q_T)$ de los estados Q es el cálculo:

$$P(O|\lambda) = P(O|q,\lambda)P(q|\lambda) = \pi_{q1} b_{q1}(o_1) a_{q1q2} \dots a_{qT-1qT} b_{qT}(o_T)$$

Esta expresión tiene un alto costo en cómputo debido a las numerosas rutas en una arquitectura exponencial, incluso es demasiado demandante para una arquitectura de Baki. Para optimizar este proceso se utiliza el algoritmo forward-backward.

- Algoritmo forward

Definiendo $\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = i|\lambda)$ la probabilidad de la secuencia de observación parcial en el estado i hasta el tiempo t, usando el modelo λ. El algoritmo general se resume de la siguiente manera.

i. Inicio:

$$\alpha_1(i) = \pi_i b_i(o_1); \qquad 1 \le i \le N$$

ii. Inducción matemática:

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(o_{t+1}) ; \qquad 1 \le j \le N, 1 \le t \le T-1$$

iii. Terminación

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

La complejidad del algoritmo se reduce considerablemente en cuanto al número de multiplicaciones y el número de sumas, en números el orden es de N^2T.

- Algoritmo backward

El algoritmo es el inverso al algoritmo de forward. $\beta_t(i) = P(o_1 o_2 \dots o_t, q_t = i|\lambda)$ es la probabilidad de la secuencia de observación parcial desde $t+1$ hasta el final, dado el estado i en el tiempo t y el modelo λ.

i. Inicio

$$\beta_T(i) = 1; \qquad 1 \le i \le N$$

ii. Inducción

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1}) \ ; \qquad t = T-1, T-2, \dots 1; 1 \le i \le N$$

iii. $\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_i \ ;$

Los cálculos necesarios para obtener β es de N^2T operaciones.

5.4.2 Algoritmo de Viterbi para decodificación

La ruta de estados más probable es útil para el aprendizaje y para el alineamiento de las secuencias del modelo, sin embargo la ruta más probable $q = (q_1, q_2, \dots, q_T)$ no es una solución única pues está depende del criterio utilizado para elegir la mejor secuencia. "El problema consiste en encontrar una secuencia $I = i_1, i_2, \dots, i_3$ tal que la probabilidad de ocurrencia de la secuencia O de esta secuencia de estados sea mayor que cualquier otra secuencia de estados. En otras palabras se busca maximizar $P(O|I, \lambda)$" (Navarrete, 2007, p.50).

La solución de esta ecuación se hace usando el algoritmo de Viterbi. El algoritmo de Viterbi es un algoritmo iterativo en el que se define una variable $\delta_i(t)$ como la probabilidad máxima de generación de una secuencia de $símbolos$ sobre cualquier secuencia simple de estados.

$$\delta_t(j) = \max_{q1, q2, \dots q(t-1)} P(q_1, q_2 \dots q_{t-1}, q_t = i, o_1 o_2 \dots o_{t-1} o_t \,|\lambda)$$

Para conocer la secuencia de estados es necesario almacenar los valores del argumento que maximizan $\delta_i(t)$. Para ello se utiliza una matriz en la que cada elemento $\psi_{it} \equiv \psi_i(t)$ contiene en índice del estado que maximiza la expresión anterior en el tiempo t. El algoritmo es el siguiente:

i. Inicialización

$$\delta_1(i) = \pi_i b_i(o_1)$$
$$\psi_1(i) = 0$$

Para 1 ≤ i ≤ N

ii. Inducción

$$\delta_t(j) = \max_{i \le i \le N} [\delta_{t-1}(i) a_{ij}] \, b_j(o_t)$$
$$\psi_t(j) = arg \max_{i \le i \le N} [\delta_{t-1}(i) a_{ij}]$$

Para 2 ≤ t ≤ T, 1 ≤ j ≤ N

iii. Terminación

$$P^* = \max_{i \le i \le N} [\delta_T(i)]$$
$$Q_T^* = arg \max_{i \le i \le N} [\delta_T(i)]$$

iv. Secuencia de estados

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

Para t=T-1,T-2,…1

El algoritmo de Viterbi está basado en el método de la programación dinámica. La estructura de retículo implementa los cálculos del algoritmo.

5.4.3 Algoritmo de Baum-Welch

El problema más difícil es determinar un método de ajuste de parámetros para la optimización del modelo y pueda satisfacer los criterios del mismo. “El objetivo de la solución a este problema es entrenar al MOM usando una secuencia de observaciones de modo que si se presentara otra secuencia con características similares después, la pueda reconocer” (Navarrete, 2007, p.53); la solución que se adopta generalmente se basa en el reajuste iterativo de los parámetros del modelo hasta alcanzar un óptimo local para la probabilidad $P(O|\lambda)$. Este método se conoce como el algoritmo de Baum-Welch.

El algoritmo Baum-Welch garantiza la convergencia uniforme hacia un máximo local de la función de probabilidad de generación. El algoritmo realiza en cada iteración una estimación del conjunto de parámetros y luego maximiza la probabilidad de generar los datos de entrenamiento utilizando el modelo, de tal modo que la nueva probabilidad es mayor o igual a la previa, los valores se asignan de acuerdo a la siguiente fórmula.

Define dos nuevas variables

$$\gamma_t(i) = P(q_t = S_i|O, \lambda)$$

“Representa a la probabilidad de estar en el estado S_i en el instante t dados la secuencias de observación O y λ”. (Navarrete, 2007, p.51).

Y

$$\xi(i,j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda)$$

“Representa a la probabilidad de estar en el estado S_i en el instante t y estado S_j en el instante $t+1$ dados la secuencias de observación O y λ”. (Navarrete, 2007, p.55).

Partiendo de estas nuevas variables se deduce lo siguiente:

Número esperado de transiciones desde S_i

$$\sum_{t=1}^{T-1} \gamma_t(i)$$

Número esperado de transiciones de S_i hasta S_j

$$\sum_{t=1}^{T-1} \xi_t(i,j)$$

Usando estos conceptos se obtienen las reestimaciones

$$\pi'_i = \text{ Frecuencia esperada del estado } S_i \text{ en } t = 1$$

$$\pi'_i = \gamma_t(i)$$

$$a'_{ij} = \frac{número\ esperado\ de\ transiciones\ de\ S_i\ a\ S_j}{número\ esperado\ de\ transiciones\ de\ S_i}$$

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$b'_j(i) = \frac{número\ esperado\ de\ visitas\ de\ S_i\ y\ observar\ v_k}{número\ esperado\ de\ visitas\ a\ S_j}$$

$$b'_j(i) = \frac{\sum_{t=1 \cap Ot=vk}^{T-1} \gamma_t(i)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

En términos de los A y B

$$\pi'_i = \frac{\sum_{j=1}^{N} \alpha_i(1) a_{ij} b_j(o_2) \beta_j(2)}{\sum_{i=1}^{N} \alpha_i(t) \beta_i(t)}$$

$$a'_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_i(t) a_{ij} b_i(o_{t+1}) \beta_i(t+1)}{\sum_{t=1}^{T-1} \alpha_i(t) \beta_i(t)}$$

$$b'_i(k) = \frac{\sum_{t=1}^{T} \alpha_i(t) \beta_i(t) \delta_k(v_k, o_t)}{\sum_{t=1}^{T} \alpha_i(t) \beta_i(t)}$$

El modelo reestimado como $\lambda' = (A', B', \pi')$ pude sugerir dos posibilidades, la primera que se llegue a un punto crítico donde $\lambda = \lambda'$, y la segunda donde λ' es más probable que el modelo λ, y se ha conseguido un nuevo modelo a partir del cual las secuencias observables son mucho más probables.

5.5 Implementación

Durante la implementación de un Modelo Oculto de Markov se deben tomar en cuenta varias características que permitirán obtener secuencias óptimas y reales.

a) Estimación inicial de parámetros. Rabiner (1989) establece algunos criterios de selección de valores "para π y para *A* se pueden obtener de valores aleatorios o uniformes" para el caso de *B* se pueden obtener "por segmentación manual de secuencias de observación, por segmentación máxima de probabilidad o por segmentación de k-promedios" (p.273).
b) Falta de datos de entrenamiento. Para modelos ocultos de Markov es necesario contar con secuencias de datos suficientes que puedan restimar valores de parámetros precisos. Rabiner, 1989, también provee soluciones para esto "aumentar el tamaño de secuencias de entrenamiento, reducir el tamaño del modelo, establecer restricciones a los parámetros (p. 274)"
c) Selección del modelo. Para ello se debe seleccionar el tipo, el tamaño, el conjunto de símbolos, la distribución, y no existe una manera correcta de hacerlo teóricamente. Además de él depende la selección de los parámetros.

CAPÍTULO 6

PRE CONFIGURACIÓN GENERAL DEL SISTEMA

6.1 Esquema general para la adquisición de información

Para la extracción de información que será usada como vector de entrada en el reconocimiento de patrones es necesario procesar las muestras de música para extraer los parámetros necesarios para el entrenamiento y la evaluación.

Para la extracción y generación del vector de parámetros he decidido usar el proceso sugerido por Salcedo (2005, p.138):

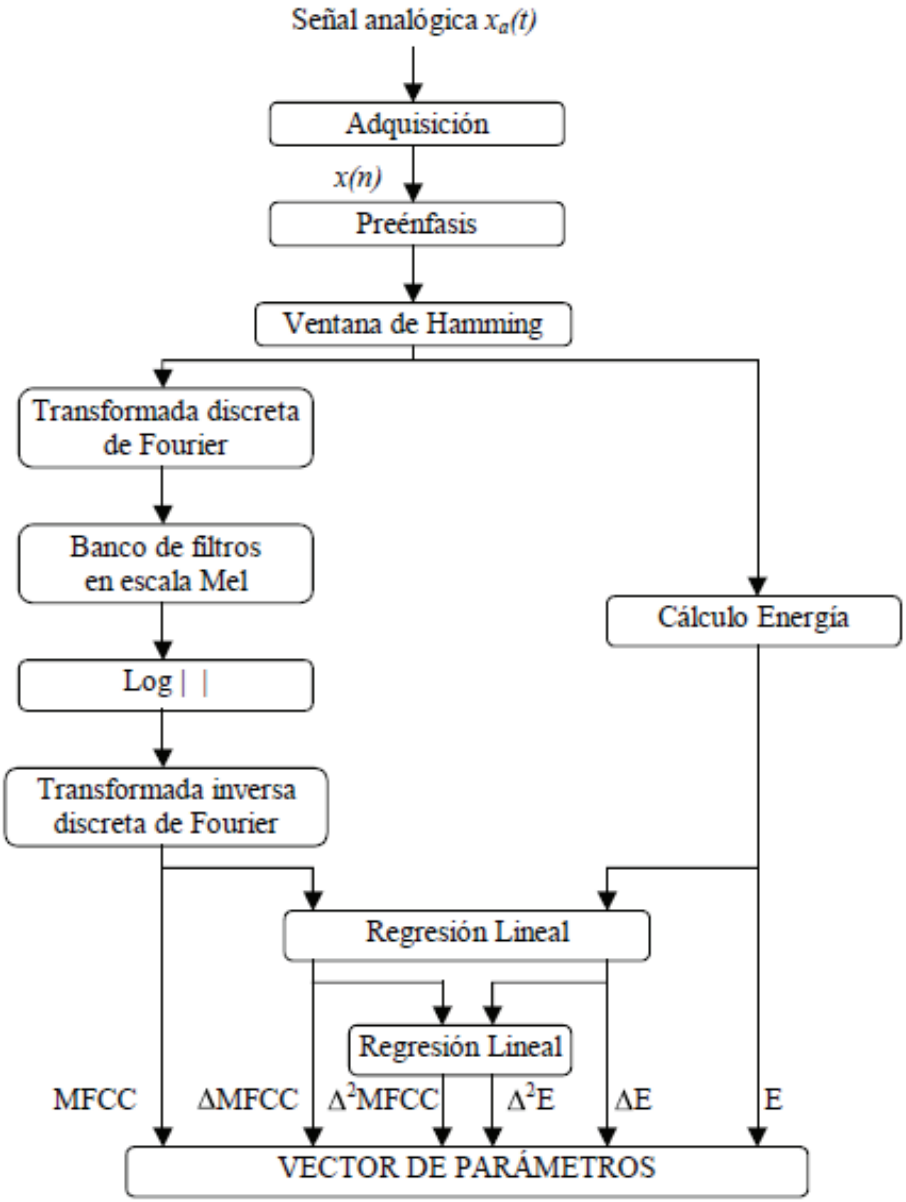


Fig. 6.1 Diagrama de flujo para la extracción de parámetros propuesto por Salcedo.

La figura 6.1 especifica las etapas del proceso para la extracción de parámetros de la señales.

La entrada es la señal analógica propiamente, en este caso la señal no ha procedido de una grabación directamente analógica, sino de los archivos MIDI generados por notas aleatorias con NOTE2MID y las composiciones modificadas en el capítulo anterior.

La adquisición de datos se refiere al proceso de transformación hacia una señal discreta. El proceso es muestreo, cuantización, codificación y compresión; ya descritos en el capítulo 2. Sin embargo aunque ya se han definido con anterioridad si cabe mencionar cuales fueron los valores usados durante este proceso:

Durante la etapa de muestreo, se uso el Teorema de Nyquist. Con el fin de obtener la frecuencia de muestreo, fue necesario crear una correlación de las notas con sus respectivas frecuencias fundamentales utilizadas, esto con el fin de asegurar que la f_m utilizada cubriera cada una de las notas y sus primeros armónicos. Tomando como base que en la escala Franco-Belga el La_3 tiene una frecuencia fundamental de 440[Hz], y usando el sistema de afinación Pitagórico obtenemos la frecuencia fundamental para cada nota (ver figura 6.2).

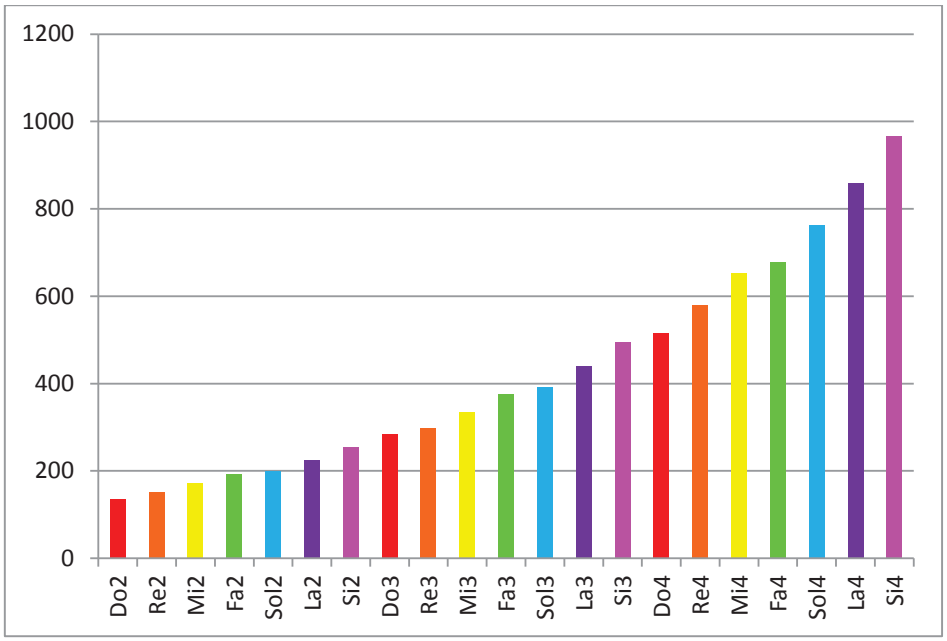


Fig. 6.2 Frecuencia fundamental de cada nota, estimada a partir de la escala franco-Belga con La_3 igual a 440 [Hz].

El rango de frecuencia va de Do_2 con 135 [Hz] a Si_4 con 966 [Hz]. De acuerdo al Teorema de Nyquist la frecuencia de muestreo debe ser al menos el doble de la frecuencia mayor que se quiera muestrear (Nyquist-Shannon, 1949).

Considerando esto, f_N debe ser al menos de 2 [kHz], sin embargo para tener un amplio rango de armónicos utilizaremos 22.5 [kHz], también consideramos que debe ser potencia de 2.

Para la cuantización no existe mayor problema, es uniforme para todos los archivos creados. En la última etapa, para la codificación, utilizo 16 bits como máximo, fácilmente comprobable utilizando un medidor de bit de profundidad.

6.1.1 Preénfasis

Es un proceso que permite realizar un filtrado para compensar la caída del espectro a altas frecuencias. Se utiliza un valor estándar de 0.97 en el pre procesado de muestras.

“La señal digital se hace pasar por un sistema de primer orden para aplanar su espectro, permitiendo que la codificación sea más uniforme” (Aguilera P.,(2007), p. 18). Este sistema tiene un solo polo en $z = \alpha$, siendo α un número cercano a 1, para que no se distorsione demasiado la señal.

$$x'(n) = x(n) - \alpha \cdot x(n-1)$$

6.2 Segmentación de la señal

"La segmentación consiste en agrupar en una ventana o segmento un número predefinido de muestras consecutivas" (Salcedo F. J., 2005, p.133). Durante el proceso de segmentación es posible utilizar el solapamiento de ventanas para simular que la señal es estacionaria.

El proceso consiste en multiplicar la señal $x[n]$ por la ventana $w[n]$ que contiene las muestras del segmento en cuestión.

La ventana se define como:

$$w[n] = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N-1}\right), n\epsilon[0, N-1]$$

Para el desplazamiento $\Delta\omega$ es común utilizar la mitad del tamaño de la ventana.

$$\Delta\omega = \frac{\Delta v}{2}$$

Para una ventana Hamming con $N = 2048$ muestras aplicadas a una señal usando un desplazamiento $\Delta\omega = 1024$ muestras. Y empleando $f_m = 22.05\ [kHz]$, la ventana tiene una duración de aproximadamente 0.1 [seg.] con un desplazamiento de 0.05 [seg.].

6.3 Extracción de parámetros

La selección de los parámetros adecuados para la detección de características musicales no es sencilla; existen muchos parámetros que han sido usados satisfactoriamente en el análisis acústico y musical, sin embargo la mayoría de estos parámetros solo son capaces de detectar patrones de una característica acústica o musical. Para está aplicación se usan parámetros con una gama un poco más amplia de utilidades.

Durey y Clements en adición a los ya mencionados utilizan la frecuencia fundamental de cada nota y sus correspondientes armónicos, además incorporan el uso de pitch para el reconocimiento de melodías concretas.

6.3.1 Escala basada en FFT

El conjunto de coeficientes de FFT conserva una gran cantidad de información que es irrelevante para el proceso orientado a la música y quizás es incluso perjudicial para él. “En un esfuerzo para descartar mucha de esta información innecesaria como sea posible, manteniendo como información relevante tanta información musical como podamos, desarrollamos un conjunto de características adicionales basado en la FFT” (Durey A., 2003, p. 58).

El método consiste en aplicar una máscara a los datos de la FFT para conservar sólo las frecuencias fundamentales y armónicas de las notas de la gama de interés.

“El proceso de enmascaramiento reduce el tamaño del conjunto de coeficientes de la FFT aproximadamente la mitad de la longitud” (Durey A., 2003, p.58).

6.3.2 Banco de filtros de la escala Mel

La FFT, captura una vista general del contenido de la frecuencia de la señal, pero también incorpora el rango perceptible por el sistema auditivo humano.

“Las frecuencias más bajas están distanciadas por espacios más estrechos, los filtros se estrechan a medida que aumenta la frecuencia, así se crea el filtro en tamaño y espaciado” (Durey A. 2003, p.60).

La figura 6.3 muestra un banco de filtros Mel.

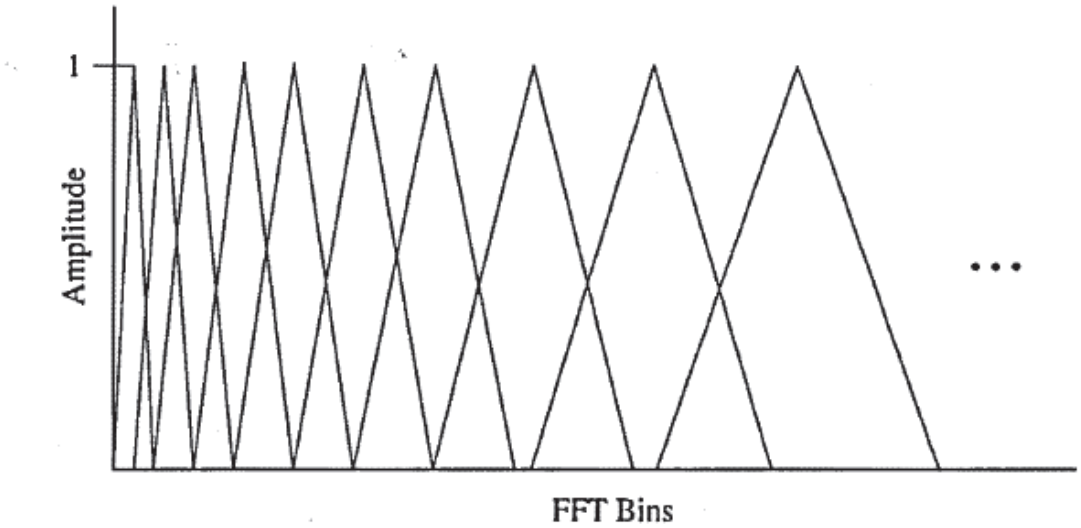


Fig. 6.3 Banco de filtros triangulares MEL.

Los canales se calculan en el rango de interés aplicando los filtros triangulares a la amplitud del registro de la FFT para cada ventana.

La expresión analítica del filtro es el siguiente:

$$H_m(k) = \begin{cases} 0\,, & k < f(m-1) \text{ ó } k > f(m+1) \\ \dfrac{2(k - f(m-1))}{(f(m+1) - f(m-1))(f(m) - f(m-1))}, & f(m-1) \le k \le f(m) \\ \dfrac{2(f(m-1) - k)}{(f(m+1) - f(m-1))(f(m+1) - f(m))}, & f(m) \le k \le f(m+1) \end{cases}$$

“El banco de filtros Mel requiere aproximadamente de tres segundos de cálculo por canción usando HTK” (Durey A., 2003, p.60).

6.3.3 MFCC

“Los coeficientes cepstrales de frecuencia Mel han sido el coeficiente dominante, usado para el reconocimiento vocal por mucho tiempo… Este suceso es debido a su capacidad de representar un espectro amplio en forma compacta” (Logan B., 2000, p.3).

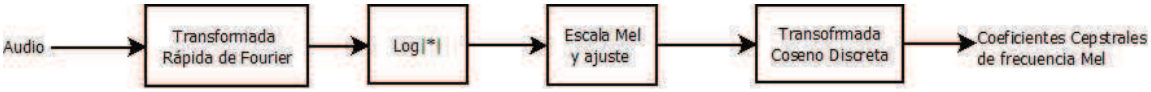


Fig. 6.4 Procedimiento para el cálculo de MFCC.

El proceso para calcular los coeficientes cepstrales de frecuencia Mel se representa en la figura 6.4 y se describe a continuación:

Antes de iniciar el proceso es necesario fragmentar en ventanas la señal original, usualmente se hace mediante Hamming en intervalos solapados.

El primer paso es tomar la FFT de cada ventana, el siguiente paso es obtener solo el logaritmo de la amplitud del espectro. “Descartamos la información de la fase porque estudios perceptuales han demostrado que la amplitud del espectro es mucho más importante que la fase. Tomamos el logaritmo de la amplitud del espectro por que el volumen percibido de una señal es aproximadamente logarítmica” (Logan B., 2000, p. 4).

El siguiente paso consiste en ajustar el espectro y enfatizar las frecuencias significativamente más perceptibles. Esto se logra aplicando el banco de filtros Mel junto con la escala Mel.

“La escala Mel se basa en una correlación entre la frecuencia real y el tono percibido como aparentemente lo haría el sistema auditivo humano” (Logan B., 2000, p. 5). La correlación es aparentemente lineal por debajo de 1 [kHz] y por encima se vuelve logarítmico (ver figura 6.5).

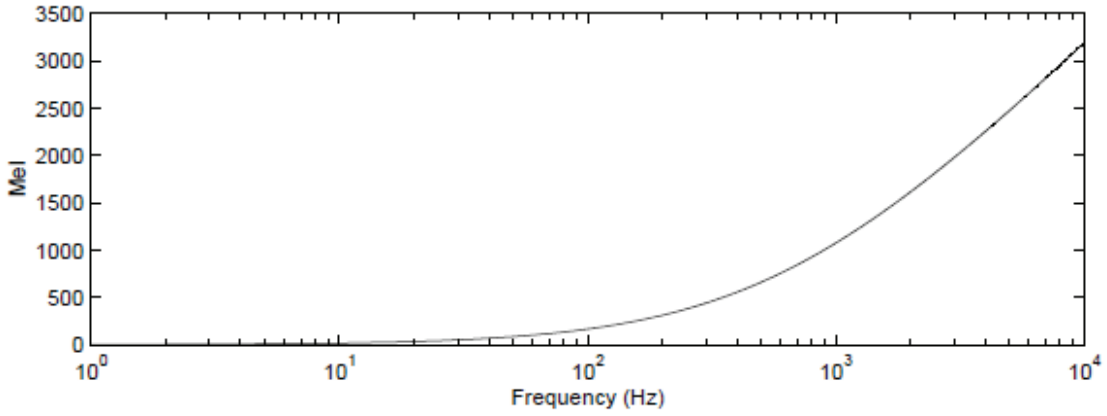


Fig. 6.5 Correlación entre la frecuencia real y la escala MEL.

Finalmente se calcula la Transformada Coseno Discreta.

Desde el punto de vista de la música, los primeros pasos son menos controversiales, al igual que la voz es no estacionario (paso 1), e independiente de la fase en grabaciones monofónicas (paso 2). Sin embargo la música tiene un rango más dinámico que la voz, pero esto no es razón para sospechar que el volumen no sea logarítmico (paso 3).

Aunque es posible que la escala Mel pueda no ser la óptima para las señales musicales, esto por la información se concentra mayormente en las frecuencias más altas (paso 4). Y finalmente el uso de la Transformada Coseno Discreta es una aproximación general de la transformada KL para la música, pero experimentos aseguran que puede ser apropiada (paso 5).

6.3.4 Energía

"La energía de una señal se define como la suma de los valores cuadráticos de la señal".

$$E = \sum_{n=0}^{N} |x[n]|^2$$

HTK calcula el logaritmo de la energía por cada ventana:

$$E = log \sum_{n=0}^{N} |x[n]|^2$$

6.3.5 Coeficientes delta y de aceleración

Para que el sistema disponga de información acerca de la evolución de la señal, se pueden incorporar coeficientes dinámicos en el vector de características. Estos coeficientes capturan parte de la correlación existente entre los vectores.

Para calcular los coeficientes delta se utiliza la siguiente formula:

$$\delta_m(t) = \frac{\sum_{n=1}^{N} n(c_m(t+n) - c_m(-n+n))}{2 \cdot \sum_{n=1}^{N} n^2}$$

N es el número de segmentos sobre los que se calculan los coeficientes dinámicos y c_m es el parámetro sobre el cual se está calculando su coeficiente de primer orden. Para el coeficiente de segundo orden o de aceleración se ocupa la misma ecuación sobre los nuevos valores o mediante una regresión lineal.

CAPÍTULO 7

CONSTRUCCIÓN y DESARROLLO DEL SISTEMA

Antes de comenzar con el desarrollo de los reconocedores es necesario resolver algunos aspectos respecto a la parametrización y al modelado de los Modelos Ocultos de Markov. Surgen algunas interrogantes:

i. ¿Cuáles son los mejores coeficientes a utilizar?
ii. ¿Cuál es el desplazamiento de las ventanas más apropiado para la extracción de coeficientes?
iii. ¿Cuántos coeficientes deben extraerse por muestra?
iv. ¿Cuál es la mejor topología a utilizar, sus estados y transiciones?
v. ¿Cómo debe conformarse la gramática?

Algunas cuestiones ya han sido respondidas anteriormente, otras se analizaran más a fondo en su momento, sin embargo no se presentaran extensos experimentos para determinar cuál es el mejor, sino serán tomadas algunas referencias de otro proyectos para resolver estas interrogantes.

También es indispensable organizar los espacios de trabajo, ordenar la metodología y establecer un marco de control para asegurar que los resultados obtenidos sean satisfactorios.

7.1 Construcción del sistema

Una vez, habiendo creado y transformado todos los archivos de datos, es necesario pasar a la etapa de construcción del reconocedor.

Antes de proseguir con cada fase del reconocedor y aplicar las funciones apropiadas para generar un resultado, es necesario preparar el espacio de trabajo para cada experimento, con el único propósito de evitar conflictos entre datos, manejo inadecuado de archivos y lecturas erróneas en los resultados.

Los siguientes procedimientos han sido extraídos del tutorial básico de HTK de Nicolas Moreau como referencia para preparar el entorno de trabajo y evitar problemas futuros.

7.1.1 Pasos para la construcción

1) Creación de la base de datos de entrenamiento y de pruebas. “Cada elemento se graba varias veces y se etiqueta con su correspondiente nombre para crear el corpus de entrenamiento” (Moreau, 2002, p.4). El proceso se encuentra ilustrado en la figura 7.1.

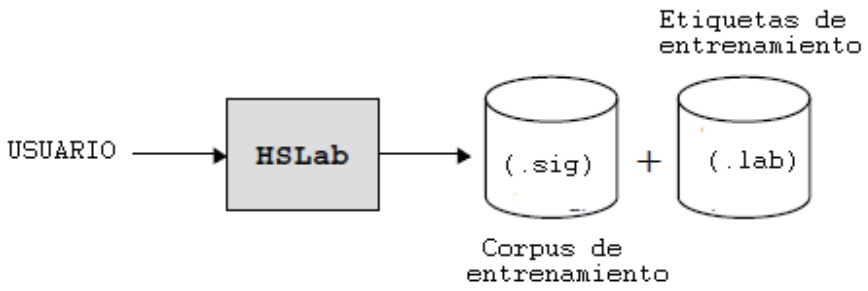


Fig. 7.1 Grabación y etiquetado de los datos de entrenamiento.

2) Análisis acústico. “Las herramientas para el reconocimiento no es capaz de procesar ondas directamente. Estas tienen que ser representadas en una forma más compacta” (Moreau, 2002, p.5). Las señales deben ser convertidas en vectores de coeficientes. Para eso se ocupa HCopy que convierte el corpus de entrenamiento a un conjunto de coeficientes (ver figura 7.2).

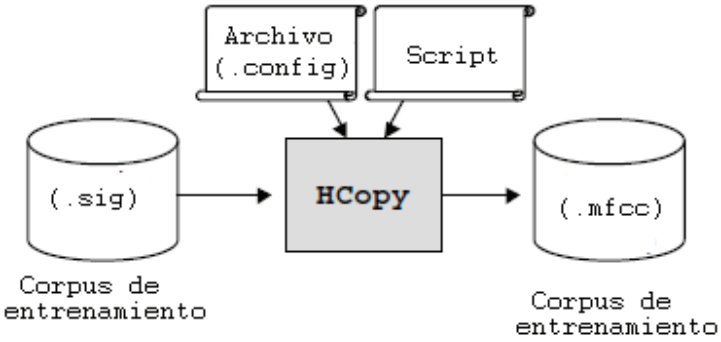


Fig. 7.2 Conversión de señales de datos a coeficientes Mel-cepstrum.

Este proceso debe realizarse tanto para las señales de entrenamiento como para las de prueba.

3) Definición de modelos. Definir un prototipo de HMM para cada elemento del vocabulario. Para este caso se utilizó la topología ilustrada en la figura 7.3.

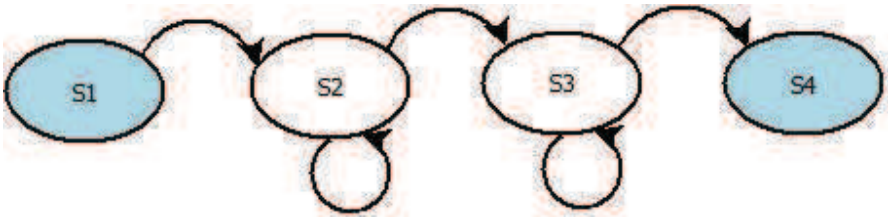


Fig. 7.3 Topología Bakis como modelo principal.

i. Elegir la topología general para los modelos.
ii. Número de estados.
iii. Forma de la función de observación asociada a cada estado.
iv. Transiciones entre estados.

4) Entrenamiento de los modelos. Cada HMM es inicializado y entrenado con los datos de entrenamiento. La figura 7.4 muestra las etapas del procedimiento del entrenamiento.

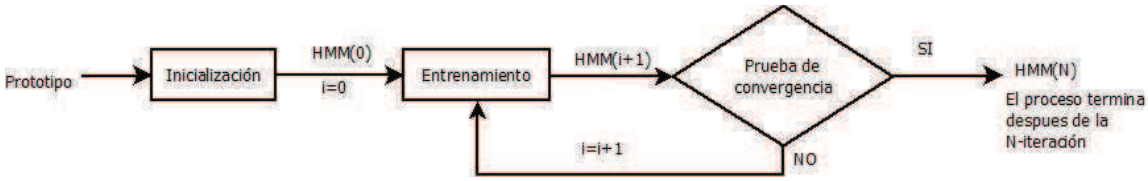


Fig. 7.4 Procedimiento completo del entrenamiento.

La primera fase del entrenamiento es la inicialización. “Alinea en el tiempo los estados de cada HMM con los datos de entrenamiento, usando el algoritmo de Viterbi” (Moreau, 2002, p.10). El diagrama 7.5 describe el proceso de la herramienta HInit para la inicialización de los modelos

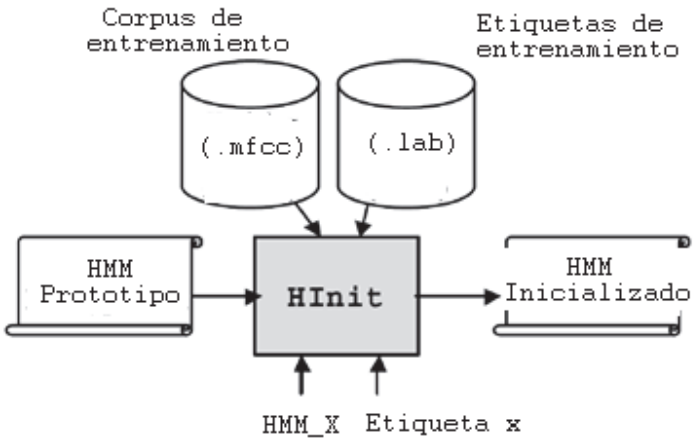


Fig. 7.5 Inicialización de HMM.

La segunda fase es la restimación. “La herramienta HRest estima los valores óptimos de los parámetros de los HMM (probabilidades de transición y vectores de medias y varianzas de cada función de observación). El diagrama 7.6 describe el funcionamiento. Cada vez que se ejecuta, producirá varias iteraciones del método de Baum-Welch hasta que converja” (Moreau, 2002, p.11).

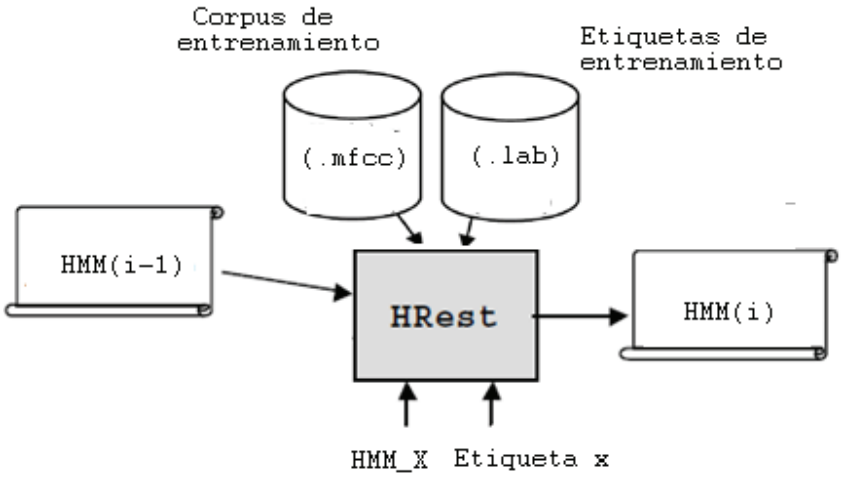


Fig. 7.6 Proceso de re-estimación.

5) Definición de la gramática. Se establece la gramática.

6) Reconocimiento de una señal desconocida. El diagrama 7.7 describe las etapas para el reconocimiento, siendo clave Hcopy y HVite, este último utiliza más recursos y archivos para el reconocimiento, además los HMM ya entrenados.

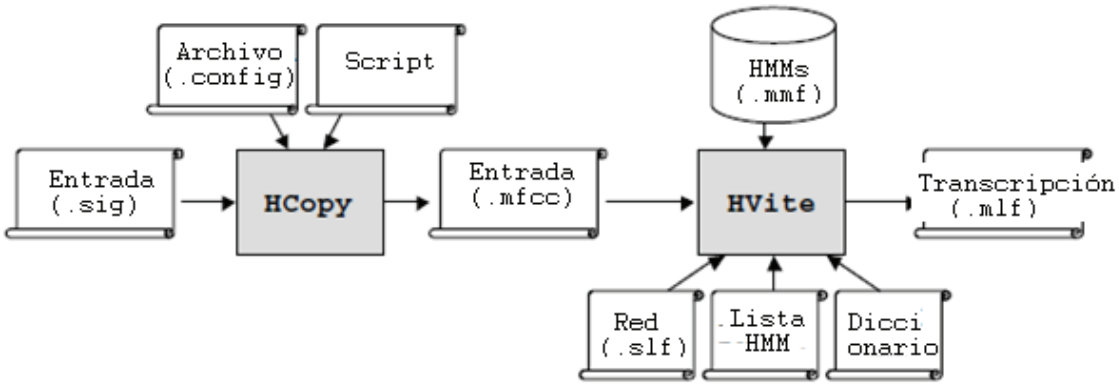


Fig. 7.7 Proceso de reconocimiento de una señal desconocida.

7) Evaluación. Se evalúa el reconocedor con los datos de pruebas.

7.1.2 Organización del espacio de trabajo

Es recomendable crear una estructura de directorios como sigue, para cada experimento:

- `datos\`: Almacena los datos del entrenamiento.
  - `datos\señales\`: Contiene los archivos de audio.
  - `datos\etiquetas\`: Contiene los archivos de etiquetas.
  - `datos\coeficientes\`: Contiene los archivos con los coeficientes.
- `analisis\`: Archivos de configuración (.config) relacionado con el análisis acústico.
- `entrenamiento\`: Contiene una lista de secuencias de coeficientes que se utilizarán en el entrenamiento.
- `modelo\`: Contiene los modelos de HMM. Prototipos, modelos inicializados, y/o modelos entrenados.
- `definicion\`: Contiene los archivos que definen la gramática.
- prueba\: Almacenará los archivos que conciernen a las pruebas.
  - `pruebas\señales\`: Contiene los archivos de audio.
  - `pruebas\etiquetas\`: Contiene los archivos de etiquetas.
  - `pruebas\coeficientes\`: Contiene los archivos con los coeficientes.

7.2 Desarrollo del sistema

Con el fin de perfeccionar los resultados se ha dividido el sistema completo en diferentes experimentos con orden creciente de dificultad, el entrenamiento se hace de forma progresiva, se toma como modelo inicial el modelo entrenado del experimento anterior, y los datos de prueba pasan a formar parte del grupo de nuevos datos de entrenamiento.

La división de experimentos tiene el objetivo de comparar los resultados con diferentes tipos de datos de entrenamiento y de pruebas. Además de presentar una perspectiva general respecto a los diferentes niveles de complejidad en la música, pasando de notas aisladas de un solo instrumento hasta llegar a la música polifónica.

La lista de experimentos a realizar se presenta en la siguiente tabla:

Clave del experimento	Cantidad de instrumentos	Datos de entrenamiento	Datos de pruebas
E1	Mono	Notas de duración fija	Notas de duración fija
E2	Multi	Notas de duración fija	Notas de duración fija
E3	Multi	Notas de duración fija y variable	Notas de duración variable
E4	Multi	Notas de duración fija y variable	Música monofónica
E5	Multi	Notas de duración fija y variable	Música monofónica
E6	Multi	Notas de duración fija, variable y música monofónica	Música polifónica
E7	Multi	Notas de duración fija, variable y música monofónica	Música polifónica
E8	Multi	Notas de duración fija, variable y música monofónica	Música polifónica

Tabla 7.1 Lista de experimentos.

La diferencia entre E4 y E5 está en que para E4 se ocupa la mitad de los archivos de música monofónica que corresponden a los mismos instrumentos que en los archivos de notas aleatorias. En E5 se ocupan los archivos restantes dejando de lado los que corresponden a los mismos instrumentos, es decir se ocupan diferentes instrumentos a los ocupados en los archivos de notas aleatorias.

7.3 Ejemplo de reconocedor mono-instrumental de notas de duración fija

Se ocupan solo 100 archivos con notas de duración fija de un solo instrumento, para este caso se ocupa el violín. Para el entrenamiento y evaluación la distribución es de 80%/20%. La figura es semicorcheas, escala franco-belga, octavas 2, 3 y 4.

7.4 Análisis acústico

El proceso consiste en extraer los parámetros de cada archivo de audio, este proceso debe aplicarse tanto para los datos de entrenamiento como de prueba.

Para el caso del experimento 1, los archivos a ocupar son los correspondientes a un solo instrumento, 40 en notación MIDI, es decir el violín. De estos el 80% de los archivos serán usados como entrenamiento y el 20 % restante para pruebas.

Antes obtener los parámetros utilizados es necesario crear el archivo de configuración. Su estructura es la siguiente:

```
SOURCEFORMAT = WAV
TARGETKIND = MFCC_E_D_A
WINDOWSIZE = 450000.0
TARGETRATE = 90000.0
NUMCEPS = 40
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 99
CEPLIFTER = 22
```

- SOURCEFORMAT: Especifica el formato de audio.
- TARGETKIND: Especifica los coeficientes que se van a extraer, para este caso:
  - MFCC; Coeficientes Mel-Cepstrum.
  - _E; energía total de la trama.
  - _D; coeficientes delta.
  - _A; coeficientes de aceleración
- NUMCEPS: Indica el número total de coeficientes que se calcularán de coeficientes MFCC $[c_1 \dots c_{40}]$, además la energía $[c_0]$, y los coeficientes dinámicos Δc y $\Delta\Delta c$. En total 123.

- WINDOWSIZE: Indica el tamaño de la trama
- TARGETRATE: Indica el desplazamiento de la ventana, significa que habrá solapamiento.
- USEHAMMING = T: Se activa el uso de la ventana Hamming
- PREEMCOEF: Coeficiente de preénfasis $\alpha = 0.97$.
- NUMCHANS: Especifica el número de bandas en el que está dividido el filtro perceptual $H_m(k)$.
- CEPLIFTER: Es el número de muestras cepstrum que se recortan.

La conversión se hace con el comando HCopy de la siguiente manera.

```
>HCopy –C análisis\analisis.conf –S datos\lista.txt
```

El archivo lista.txt contiene una lista de archivos de entrada con sus correspondiente salida.

7.5 Definición de los HMM

Lo primero que se debe hacer, es elegir la topología que se va usar, para el caso de la notas de utilizará Bakis de 4 estados (ver figura 7.8), con una transición adicional. Este proceso debe realizarse para cada nota musical.

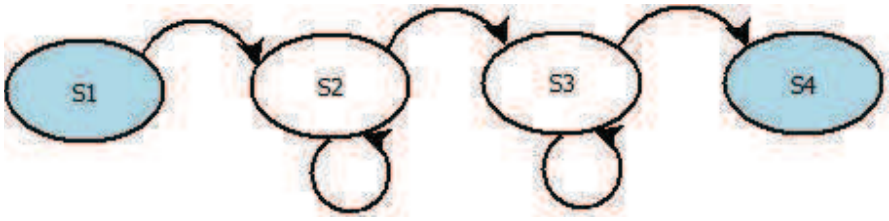


Fig. 7.8 Topología Bakis para el reconocimiento de las notas y el silencio.

La función de observación para cada estado $b_j(k)$ "es una función de densidad de probabilidad gaussiana unidimensional (con matrices diagonales no nulas), inicializados con media 0 y varianza 1. Cada elemento de la diagonal representa a un coeficiente MFCC" (Aguilera, 2007, p. 91). Esta funciona variará una vez aplicados los datos de entrenamiento.

La matriz de transiciones entre estados a_{ij}, es una matriz cuadrada, de dimensión igual al número de estados.

7.6 Entrenamiento

El entrenamiento es la parte más importante del reconocedor, porque de él dependen los resultados globales del sistema. Un mal entrenamiento, escasez de datos o un mal contenido de los mismos puede provocar graves errores.

7.6.1 Inicialización

La inicialización es el proceso que permite que el algoritmo de entrenamiento converja rápidamente.

Para la inicialización usaremos la herramienta HInit, como sigue

```
>HInit –S entrenamiento/entrenamiento.txt –M modelo\hmm0 –H
modelos\proto\hmm_nota –l xlab –L datos\etiquetas\ nota
```

Hinit utiliza el algoritmo de Viterbi. La nota se sustituye por el nombre de la nota. Entrenamiento.txt contiene una lista con los archivos .mfcc.

7.6.2 Reestimación

El siguiente paso consiste en realizar una iteración del proceso de reestimación de los modelos.

```
>HRest –T 1 –S entrenamiento/entrenamiento.txt –M modelo/hmm1
–v min_varianza –H modelo/hmm0/hmm_nota –l xlab –L datos/lab/
nota
```

La x se sustituye por el nombre de la nota y como en HInit el proceso se repite para cada nota. Min_varianza es el límite inferior para la varianza, obtenido por inspección del archivo vFloors que devuelve HCompv, este valor evita errores futuros de cálculo.

7.7 Definición de la gramática

7.7.1 Gramática

La gramática se escribe en un archivo de texto, acorde con ciertas reglas sintácticas.

```
$nota = DO2 | RE2 | MI2|… | SI4 | S
(<$nota>)
```

La variable nota puede remplazarse por DO, etc. Las llaves <>denotan que se repetirá varias veces la variable $nota. La representación gráfica de la red se encuentra en la figura 7.9.

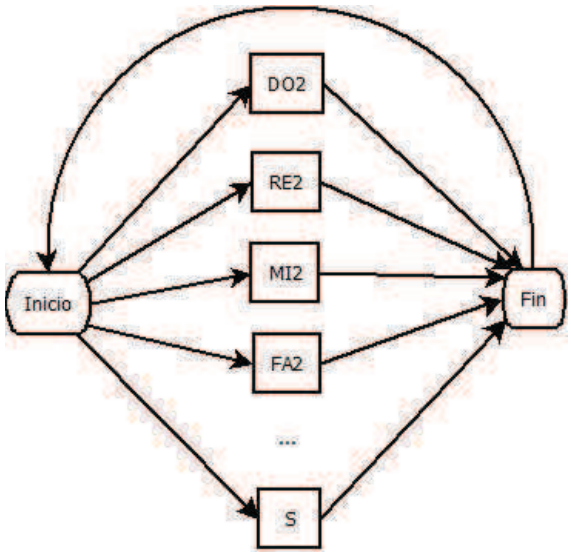


Fig. 7.9 Diseño gráfico de la gramática.

7.7.2 Diccionario

El sistema debe saber a qué HMM corresponde cada una de las variables del archivo gramática. Esta variables se almacenan en un archivo llamado diccionario.txt.

El archivo está formado por tres columnas:

a) La primera, se refiere a los nombres de las variables en la gramática.
b) La central, indica los símbolos que serán transcritos por el reconocedor.
c) La última son los nombres de los HMM.

7.7.3 Red gramatical

La gramática definida antes de usarla para ejecutar el reconocedor, debe ser compilada con la herramienta `Hparse` para obtener la red.

```
>HParse definición\gramatica.txt definición\red.slf
```

7.7.4 Comprobación de la gramática

`HSgen` permite verificar que la gramática no contiene errores, esta genera un número aleatorio de frases de acuerdo a la estructura de la gramática.

```
>HSGen –n 20 –s definición\red.slf definición\diccionario.txt
```

7.7.5 Lista de HMMs

Para el reconocimiento, es necesario crear un archivo que liste todos los HMMs disponibles, solo debe contener el nombre del HMM.

7.8 Reconocimiento

Para el reconocimiento se aplica el mismo proceso para las señales de prueba. Primeso se aplica HCopy para la extracción de parámetros.

```
>HCopy –C análisis\analisis.conf –S pruebas\listaP.txt
```

La observación de entrada se procesa mediante e algoritmo Viterbi, calculando la probabilidad de que cada HMM produzca dicha observación. Para ello se ocupa la herramienta HVite, de la siguiente manera:

```
>HVITE -T 1 -S pruebas\pruebas.txt -H modelo\hmm1\hmm_DO2 -H
modelo\hmm1\hmm_RE2 -H modelo\hmm1\hmm_MI2 -H
modelo\hmm1\hmm_FA2 -H modelo\hmm1\hmm_SOL2 -H
modelo\hmm1\hmm_LA2 -H modelo\hmm1\hmm_SI2 -H
modelo\hmm1\hmm_DO3 -H modelo\hmm1\hmm_RE3 -H
modelo\hmm1\hmm_MI3 -H modelo\hmm1\hmm_FA3 -H
modelo\hmm1\hmm_SOL3 -H modelo\hmm1\hmm_LA3 -H
modelo\hmm1\hmm_SI3 -H modelo\hmm1\hmm_DO4 -H
modelo\hmm1\hmm_RE4 -H modelo\hmm1\hmm_MI4 -H
modelo\hmm1\hmm_FA4 -H modelo\hmm1\hmm_SOL4 -H
modelo\hmm1\hmm_LA4 -H modelo\hmm1\hmm_SI4 -H
modelo\hmm1\hmm_S -i pruebas\results.mlf -w
definicion\red.slf definicion\diccionario.txt
definicion\hmmlist.txt
```

El resultado es similar a los archivos LAB.

7.9 Rendimiento del reconocedor

7.9.1 Especificar etiquetas reales

Para obtener la tasa de error del reconocedor, se debe comparar la transcripción obtenida de un conjunto de entradas de prueba, frente al etiquetado real de esas secuencias.

7.9.2 Extracción las transcripciones estimadas

Para extraer las transcripciones estimadas, es necesario ejecutar el reconocedor sobre el conjunto de pruebas. Además debe suministrarse una lista de los archivos a probar, como se hizo con la lista de entrenamiento.

Se aplica el mismo procedimiento que para el análisis acústico en la señales de prueba. Una vez hechos todos los preparativos, se ejecuta HVite.

definición\diccionarioprueba.txt es un archivo similar a definición\diccionario.txt, con la diferencia de que la segunda columna debe contener, entre corchetes, los nombres de las etiquetas asociadas a cada HMM. Esto es para que las transcripciones del reconocedor y las etiquetas reales pertenezcan al mismo diccionario.

7.9.3 Reconocimiento en tiempo real.

Para el reconocimiento en tiempo real se usa la misma herramienta, HVite, solo se agregan dos parámetros extras y es necesario crear un nuevo archivo de configuración para la entrada.

Los parámetros extras son:

- `g` Indica que debe reproducir la señal de entrada por los altavoces.

- `C análisis\directin.conf` es un archivo de configuración creado únicamente para el reconocimiento en vivo. El archivo es idéntico al del análisis acústico con tres diferencias.

```
SOURCERATE = 625.0  # 16 kHz de freciencia de muestreo
SOURCEKIND = HAUDIO # Entrada directa de audio
SOURCEFORMAT = HTK  # Formato de los archivos de voz
AUDIOSIG = -1       # Reconocimiento controlado por teclado
```

7.9.4 Tasa de error

Para la tasa de error se ocupa HResult:

```
>HResults -T 1 -L prueba\etiquetas definición\hmmlist.txt
preuba\results.mlf
```

Tras su ejecución, la consecuencia es una tabla de resultados.

```
=============== HTK Results Analysis =======================

  Date: Sat Oct 19 18:26:14 2013

  Ref: pruebas/etiquetas

  Rec : pruebas\results.mlf

------------------------ Overall Results --------------------
------

SENT: %Correct=100.00 [H=20, S=0, N=20]

WORD: %Corr=100.00, Acc=100.00 [H=4307, D=0, S=0, I=0,
N=4307]
```

La explicación de los resultados:

- Ref : es el directorio donde se encuentran las etiquetas de referencia.
- Rec: es el archivo con las transcripciones del reconocedor.
- N: es el total de transcripciones.
- H: es el número de transcripciones correctas
- S es el número de transcripciones en las que se ha sustituido la etiqueta correcta.
- Sent: %Correct =100.00 muestra el procentaje de acierto.

$$\%Correcto = \frac{H}{N} \times 100\%$$

- D: es el total de transcripciones eliminadas. Errores.
- I: es el total de transcripciones insertadas. Errores.
- WORD: %Corr=98.00, Acc=98.00 muestra el porcentaje de acierto contado en palabras.

$$\%Precisión = \frac{H - I}{N} \times 100\%$$

7.10 Resultados de los experimentos

Exp.	Entrenamiento		Pruebas		Resultados					
	#Archivos	Tipo	#Archivos	Tipo	%Correcto	%Precisión	H	S	D	I
E1	80	Notas dur. Fija	20	Notas dur. fija	100	100	4307	0	0	0
E2	400	Notas dur. Fija.	100	Notas dur. fija	80.18	76.23	17250	1430	2835	849
E3	900	Notas dur. Fija Notas dur. Var.	100	Notas dur. var.	93.64	64.29	2945	137	63	923
E4	1000	Notas dur. Fija Notas dur. var.	50	Música mono	73.73	61.36	3738	711	621	627
E5	1000	Notas dur. Fija Notas dur. var.	50	Música mono	70.77	54.64	3588	731	751	818
E6	1110	Notas dur. Fija Notas dur. var. Música mono	10	Música polifónica 2i	45.08	37.32	670	325	714	247
E7	1120	Notas dur. Fija Notas dur. var. Música mono Música poli. 2i	10	Música polifónica 3i	41.28	35.49	1132	405	1205	159
E8	1130	Notas dur. Fija Notas dur. var. Música mono Música poli. 3i	10	Música polifónica 3i	32.26	30.20	1190	418	2081	76

7.10.1 Resultados en tiempo real

```
Read 22 physical / 22 logical HMMs
Read lattice with 25 nodes / 67 arcs
Created network with 49 nodes / 91 links


READY[1]>
DO3 RE3 FA3 MI3 SOL3 LA3 DO3 ==  [4107 frames] -792.0364
[Ac=-3252893.5 LM=-255.5] (Act=47.0)


READY[2]>
SI3 LA3 SOL3 FA3 MI3 RE3 S ==  [3040 frames] -792.1517 [Ac=-
2408141.3 LM=-387.4] (Act=47.0)

READY[3]>
quit
```

CAPÍTULO 8

CONCLUSIONES

8.1 Conclusiones respecto al desarrollo

Este trabajo es el resultado de la aplicación de conceptos pertenecientes a diversas disciplinas. Algunas, fueron pieza clave en el desarrollo y construcción del sistema. El sonido, como rama de la Física, proporcionó las bases para el manejo adecuado de una señal, permitió aclarar cuáles son los componentes esenciales (timbre, altura, tono) para la extracción de parámetros de una señal musical así como su relación directa con esta.

Por otra parte la teoría musical fue la piedra angular, pues proporciona los fundamentos sólidos para la detección de patrones simples en el reconocimiento de notas musicales. Para esto fue necesario conocer la interpretación de las notas musicales en un desarrollo físico y digital, conocer la frecuencia de ellas y marcar su lugar dentro de una escala. Las notas son los patrones musicales más simples por ser el principio más sencillo de la música, de ellas surgen las melodías, el ritmo y finalmente las composiciones.

La selección de la escala musical Franco-Belga sirvió para determinar un punto de referencia adecuado. Las octavas para el reconocimiento utilizadas son las tres centrales de toda la escala. La selección de las octavas estuvo condicionada por las muestras de música polifónica, ya que estas concentran una mayor cantidad de notas en estos intervalos.

Turing estableció que no existe un algoritmo que no pueda ser implementado en una máquina. La solución a los problemas de los modelos ocultos de Markov, presentan soluciones matemáticas, fácil de implementar y ejecutar, como recetas de cocina. La solución óptima, además de implementar los algoritmos de solución, fue utilizar herramientas de ejecución, estas optimizaron los tiempos de ejecución gracias al cómputo paralelo o distribuido, para ello, la herramienta HTK fue indispensable.

8.2 Conclusiones respecto a la parametrización

La escala MEL para la obtención de los coeficientes cepstrales, ha tenido resultados favorables en la detección del habla, sin embargo ha quedado demostrado que su uso para fines musicales también es eficiente.

Para la determinación del número de coeficientes, se tomó como referencia a algunos autores, sin embargo la cantidad propuesta fue ajustada a 40, por dos razones principales; la primera, el número de coeficientes está directamente relacionado con el número de estado activos en la topología, para el caso de HTK, crean un total de coeficientes de saturación que impide generar más; y la segunda, tras una serie de pruebas complementarias se determinó que 40 era el número que mejores resultados ofrecía.

La elección del ventaneo fue lo más complicado, segmentos muy cortos aumentan considerablemente el tiempo de extracción de coeficientes, además el exceso de datos disminuye el rendimiento del reconocedor y existe la posibilidad de que interprete erróneamente los patrones dando un reconocimiento inexacto. Segmentos muy largos no consideran información que podría ser un discriminante durante el reconocimiento. Para el solapamiento se usó el 20% de la longitud de la trama y crear redundancia entre las ventanas.

El proceso de ventaneo también se diseñó considerando a los instrumentos, por ejemplo, las notas en el piano tienen una caída abrupta de energía al final, lo que podría interpretarse como un silencio, este problema queda solucionado con un ventaneo corto. El ventaneo corto también permite una mejora en el reconocimiento de notas de duración variable; los experimentos iniciaron con notas semicorcheas, por ser la de menor duración, dan la pauta para delimitar los valores mínimos en la parametrización. En el caso de notas de duración mayor el ventaneo, además de tener datos suficientes para el reconocimiento, permite obtener nuevos valores para el refinamiento de los coeficientes y en consecuencia una mejor identificación de las notas.

La topología también jugó un papel importante en la eficiencia, la selección del número de estados y las transiciones adicionales son punto clave. Para los experimentos se hicieron dos pruebas, una con topologías Bakis de 2 estados activos y otra con 4 estados activos, los resultados fueron más favorables con 2 estados. Esto debido a que las notas son más simples que una palabra y no necesitan demasiados estados, un mayor número de estados disminuye la capacidad de reconocer al estar recibiendo nuevas observaciones adicionales por cada transición.

Finalmente cabe destacar, que la necesidad de crear una topología para el silencio, es porque tras una ausencia de información en la señal el sistema puede tener dos problemas: tras un silencio largo, el sistema intentará insertar una nota donde no la hay, en cambio tras un silencio corto, el sistema puede alargar la nota anterior hasta cubrir el tiempo del silencio.

8.3 Conclusiones respecto a los resultados

El entrenamiento progresivo ofrece los mejores resultados. Tras cada experimento se efectúa un nuevo entrenamiento con los modelos finales del experimento anterior, esto ofreció mejores resultados que al realizar en cada experimento un entrenamiento completo. Al observar la gráfica 8.1 se observa los porcentajes de los diferentes resultados.

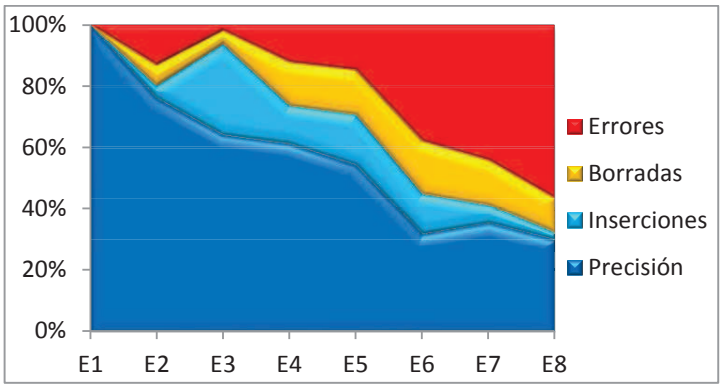


Fig. 8.1 Resultados de los experimentos con señales grabadas.

Tras cada experimento se aprecia una caída en el porcentaje de precisión.

Anteriormente se explicó que con cada experimento aumenta la dificultad para reconocer las notas. La imagen lo ilustra más claramente. El entrenamiento gradual no ha sido suficiente, aunque si causo mejorías también no fue capaz de adaptarse completamente a los nuevos obstáculos en cada experimento. Problemas que se pueden corregir incrementando significativamente los datos de entrenamiento.

A continuación se explica cada una de las complicaciones a las que se tuvo que adaptarse el reconocedor:

1. Los nuevos instrumentos introducidos. E1 solo examina un instrumento; para E2, E3 se introducen cuatro instrumentos más; E4 y E5 se incrementan a cinco instrumentos más, y para el resto se ocupan otros distintos. A pesar de esto, es posible determinar una nota por su frecuencia fundamental; la energía y amplitud varía de acuerdo al instrumento.
2. La velocidad de interpretación de las notas. Para los archivos con notas aleatorias todas manejan una velocidad constante de 100 mbps, pero para los archivos de música esta varia con cada nota y para cada instrumento.
3. La mala interpretación de los silencios, insertando notas o entendiéndolas.
4. En la música polifónica existen notas que se sobreponen, el entrenamiento se realizó usando notas aisladas; el sistema no es capaz de identificar todas las

notas solapadas ni separarlas, mucho menos determinar sus tiempos de inicio ni final.

5. Para los resultados obtenidos en tiempo real, la identificación de notas no tuvo mayor problema y se concreto en resultados positivos. Este tipo de resultados no son representados en un porcentajes, por el contrario son observables al final del reconocimiento.

APÉNDICE A

RECURSOS DE HARDWARE Y SOFTWARE

La importancia de conocer el equipo cómputo, consiste en saber cuáles son sus capacidades y limitaciones, ya que este tipo de información puede solucionar muchos problemas antes de que se presenten.

Entre algunos problemas que surgen por la mala administración del hardware, está la sobrestimación de la capacidad de la memoria principal, malos cálculos de procesamiento pueden saturar la memoria principal, volver lenta la aplicación e invadir espacios de memoria reservados. Un problema similar a éste, se presenta en los medios de almacenamiento secundario, donde los malos cálculos de tamaño por dato almacenado pueden ocasionar sobreescrituras, desbordamientos y pérdida de información.

Otros problemas ocasionados por hardware son la incompatibilidad que tienen algunos dispositivos para comunicarse entre sí, tales son los casos como dispositivos de red, medios de transmisión, interfaces, periféricos. De todos estos dispositivos se tiene que considerar velocidad de transmisión, tiempo de latencia, interferencias, calidad de transmisión, etc.

Y el elemento más importante es el procesador. Cualidades como el rendimiento, tiempo de respuesta, velocidad, la estructura de la palabra, número de núcleos; pueden generar un desastre desde entregar resultados erróneos hasta nunca entregar ningún resultado. Cada fabricante diseña su propio procesador y pequeñas diferencias pueden significar grandes cambios a la hora de ejecutar una aplicación compartida, donde la sincronización es un factor clave para un resultado óptimo.

En cuanto al software, los problemas más habituales están por la incompatibilidad de aplicaciones relacionadas con el sistema operativo, ya sea Windows o Linux, la versión de las aplicaciones, lenguajes de programación y protocolos de red deben ser compatibles con la versión del sistema operativo. El programa debe ser capaz de ejecutarse de manera transparente en todos los equipos de la red. El lenguaje de programación debe satisfacer ciertas necesidades de rendimiento, soporte, conexiones, optimización y comunicación. Las aplicaciones referentes a las bases de datos deben permitir un acceso concurrente, consistente y confiable. Los protocolos de red deben permitir una comunicación integra y confiable, deben poder manejar situaciones de errores y colisiones.

Para evitar este tipo de situaciones, es el objetivo de esta sección, dar un vistazo al hardware y software disponible para la aplicación. De ser el caso buscar soluciones baratas, buscar algún remplazo, adaptar la aplicación; siempre y cuando se encuentre un equilibrio entre las partes.

A.1 Hardware

A.1.1 Equipo

Los equipos a utilizar son dos computadoras portátiles de la marca Sony, la ventaja de estos equipos es que se trata de computadoras con hardware similar, mismo procesador, mismo sistema operativo. Las características generales del equipo, las muestro en la tabla A.1, las del procesador en la tabla A.2 y las de la memoria RAM en la tabla A.3:

Nombre de equipo	Nestor
Marca	Sony Vaio
Modelo	VGN-FZ340E
Sistema operativo	Windows 7 Home Premium
Tipos de sistema	64 bits
Procesador	Intel® Pentium ® Dual CPU
Memoria RAM	4096 MBytes
Disco duro	214 GBytes
Puertos de red	Fast Ethernet
Controladora de sonido	Realtek High Definition Audio
Conectores USB	3 conectores 2.0

Tabla A.1 Características de la computadora.

A.1.2 Procesador

Nombre	Intel Pentium T3200
Marca	Intel
Especificación	Intel ® Pentium ® Dual CPU T3200 @ 2.00 Ghz
Número de núcleos	2
Número de hilos	2
Tecnología de fabricación	65 nm
Velocidad del bus (FBS)	665 MHz
Velocidad por núcleo	1 GHz
Voltaje máx. del procesador	1.25 V
Memoria cache	Nivel 1 Datos 2 x 32 KBytes, Instr. 2 x 32 KBytes
	Nivel 2 1024 KBytes

Tabla A.2 Características del procesador.

A.1.3 Memoria RAM

Equipo	Nestor	Nestor	Gato	Gato
Capacidad	2048 MBytes	2048 MBytes	2048 MBytes	1024 MBytes
Fabricante	Kingston	Kingston	Qimonda	Kingston
Canales	Dual	Dual	Dual	Dual
Tipo	DDR2	DDR2	DDR2	DDR2

Máx. ancho de banda	333 MHz	333 MHz	333 MHz	333 MHz
Latencia CAS	3 ciclos	3 ciclos	3 ciclos	3 ciclos
Relación FBS:DRAM	1:2	1:2	1:2	1:2
Voltaje máx.	1.80 V	1.80 V	1.80 V	1.80 V

Tabla A.3. Características de la memoria RAM.

La latencia CAS es el tiempo (en número de ciclos de reloj) que transcurre entre que el controlador de memoria envía una petición para leer una posición de memoria y el momento en que los datos son enviados a los pines de salida del módulo.

A.2 Software

El sistema operativo a utilizar es Windows 7, para los equipos. La decisión es simple, es el sistema operativo que tienen los equipos por defecto. Además instalar un sistema diferente en los equipos puede aportar solo algunas ventajas, pero desequilibrar otras características, otra ventaja evitar modificar datos o crear directorios especiales para la migración de información entre distintos sistemas operativos.

APÉNDICE B

HERRAMIENTA HTK

HTK es una herramienta poderosa que se emplea para el diseño, la construcción y la manipulación de modelos ocultos de Markov (HMM), los cuales constituyen la herramienta matemática más efectiva actualmente para implementar sistemas de reconocimiento del habla.

En 1989, el profesor Steve Young del departamento de ingeniería de la Universidad de Cambridge, desarrolla este conjunto de bibliotecas. Su propósito inicial fue la investigación en reconocimiento del habla, sin embargo, al paso del tiempo se notó su potencial para aplicarse en otros ámbitos.

Al ser único requisito de uso que los problemas se modelen con HMM, puede el HTK emplearse en sistemas de reconocimiento y síntesis de voz, reconocimiento de caracteres y formas gráficas, análisis de vibraciones mecánicas e, inclusive en la determinación de secuencias de ADN.

HTK está conformado por un conjunto de bibliotecas y herramientas en forma de archivos de código fuente, escritos en C.

Las características en el presente manual y empleadas en el desarrollo de los ejemplos mostrados más adelante, son:

- Rediseño principal de muchas bibliotecas y herramientas.
- Documentación a través del “HTK Book”.
- Soporte para HMM de densidad discreta.
- Reescritura completa de herramientas de reconocimiento empleando un nuevo formato gramatical basado en retículos (redes, lattice) (módulos HNet/HRec).
- Soporte para trifonemas de palabras cruzadas, lattice y n-mejor reconocimiento de salida y “back-off” modelos de lenguaje bigrama.
- Agrupación de estados de árboles de decisión. (Árboles de decisión basados en la agrupación de estados).
- Entradas de audio y habla rediseñadas (HWave/HParm) para soportar cohersión desde las formas de onda y audio de entrada en tiempo real.
- Archivos de configuración.

“Un sistema de reconocimiento del habla, desde el inicio hasta el final, está formado por una serie de tareas que garantizan la obtención de un corpus de buena calidad, por medio del cual se entrena al sistema para, en el reconocimiento, conseguir cierto grado de aceptación” (Aparicio, 2011, p.4).

Así, lo primero que debe ejecutarse es la definición del Corpus de Voces, esto es, qué palabras o frases se desean reconocer; acto seguido, se establecen condiciones adecuadas para realizar su grabación y ésta se lleva a cabo. Posteriormente debe realizarse un preprocesamiento de la señal, para pasar a la extracción de parámetros de las señales de voz. El entrenamiento es la etapa siguiente y, por último, se verifica el porcentaje de reconocimiento.

B.1 Herramientas de HTK

B.1.1 Arquitectura.

Gran parte de la funcionalidad de este programa se debe a que está constituido con módulos de bibliotecas. Mediante el uso de dichos módulos se garantiza que la interacción de las aplicaciones con el exterior funcione siempre igual. Ellos también proveen de un recurso central de las funciones empleadas con frecuencia.

El modelo de desarrollo de una aplicación y su comportamiento, consiste en un núcleo que permanece rodeado por las “herramientas” HTK empleadas (ver figura B.1).

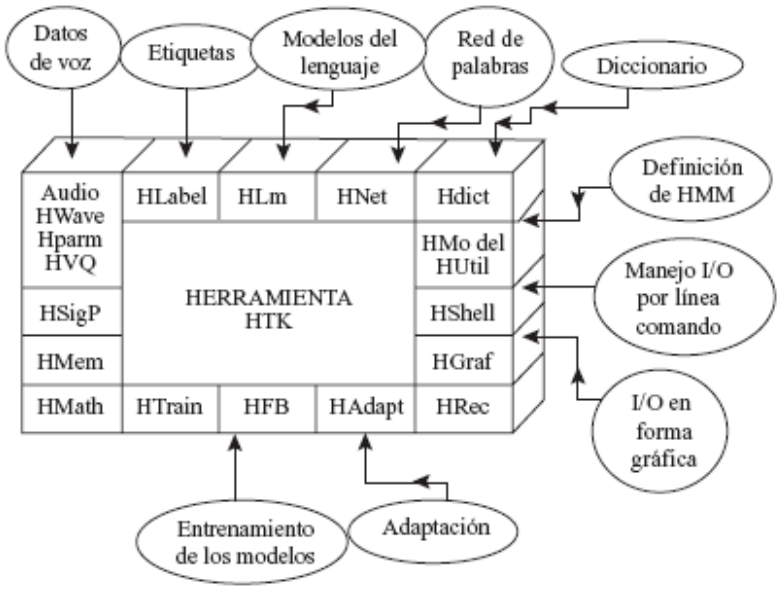


Fig. B.1 Arquitectura de las herramientas HTK.

Con base en el modelo previamente mostrado se da una descripción de algunas herramientas:

- HAdapt: Soporte para herramientas de adaptación de HTK.
- HAudio: Soporte para entradas directas de audio.
- HDict: Se usa con los diccionarios de la aplicación.
- HFB: Soporte para herramientas de entrenamiento de HTK.
- HGraf: Gráficos interactivos simples.
- HLabel: Provee la interfaz para archivos de etiqueta.

- HLM: Provee la interfaz para archivos de modelo de lenguaje.
- HMath: Soporte matemático.
- HMem: Manejo de memoria.
- HModel: Para definiciones de HMM.
- HNet: Provee la interfaz par redes y retículos.
- HParm: Trabajo a nivel de parámetros.
- HRec: Contiene las principales funciones para reconocimiento.
- HShell: Entrada/salida con el usuario e interacción con el SO.
- HSigP: Operaciones de procesamiento de señales para reconocimiento del habla.
- HTrain: Soporte para herramientas de entrenamiento de HTK.
- HUtil: Contiene un número de rutinas de utilidad para el manejo de HMM.
- HVQ: Se usa con libros de código VQ.
- HWave: Se emplea para las entradas y salidas en el nivel de forma de onda.

B.2 Propiedades genéricas de una herramienta HTK

Las herramientas HTK están diseñadas para funcionar desde un entorno de línea de comandos. Éstos, para su escritura requieren parámetros, los cuales pueden ser necesarios u opcionales; para su escritura necesitan estar precedidos por un guión.

Las letras de los comandos opcionales pueden ser mayúsculas o minúsculas; "aquellas que se encuentran en mayúscula corresponden a opciones que tienen significado para todas las herramientas de HTK; por su parte, las minúsculas corresponden a una opción con significado propio a cada herramienta" (Aparicio, 2011, p.7).

Existe la posibilidad de guardar parámetros o valores de configuración de la aplicación a través de archivos y tomarlos de ellos a partir de una llamada al archivo en la misma línea de comando, por medio de -C Nombre_Archivo, como se muestra a continuación, en donde los valores de configuración se toman del archivo llamado "config".

HFoo -C config -f 34.3 -a -s myfile file1 file2

Sin embargo, no sólo de un archivo pueden leerse dichos valores; es posible realizar consultas de éstos a partir de más de un archivo, como se aprecia en la siguiente línea, siendo "config1" y "config2" los archivos de configuración leídos.

HFoo -C config1 -C config2 -f 34.3 -a -s myfile file1 file2

Los parámetros de configuración pueden ser empleados como alternativa a la escritura en línea de comandos, pero su principal uso es controlar el comportamiento detallado de los módulos de biblioteca de los cuales dependen todas las herramientas HTK y la ejecución de las mismas.

Es posible observar las opciones de configuración de cualquier herramienta si se escribe en línea de comandos su nombre únicamente. Tomemos el ejemplo de HCopy, en la figura B.2:

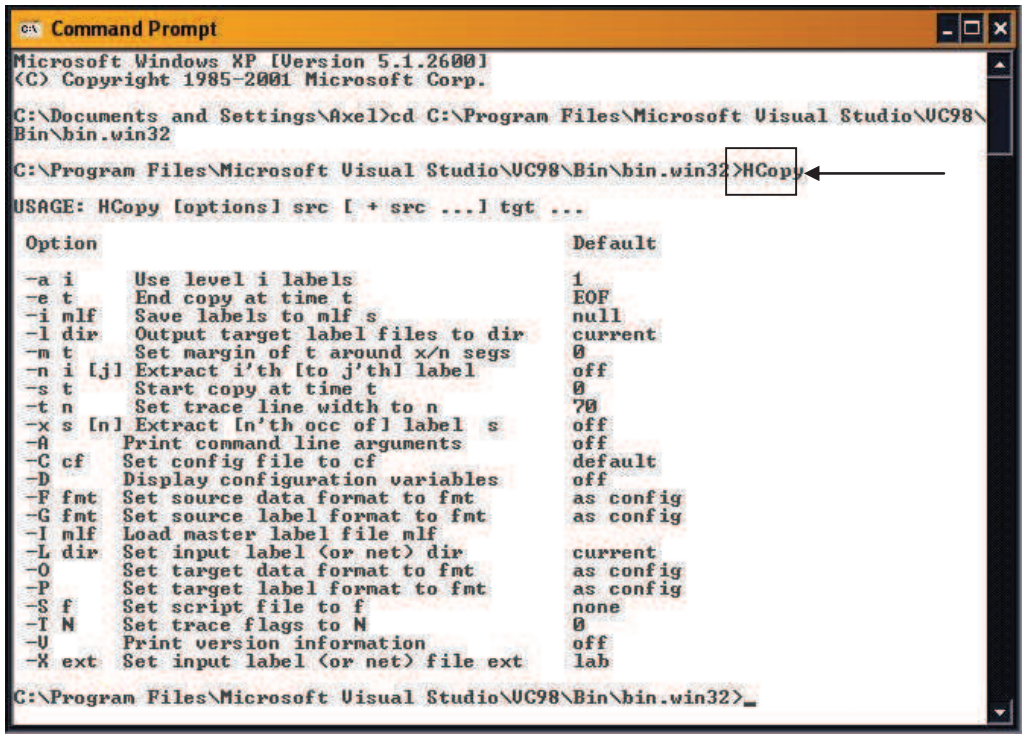


Fig. B.2 Opciones del comando HCopy.

La forma general de los comandos escritos es: *herramienta [opciones] archivos*.

Las opciones pueden ser variables, parámetros del sistema y/o bibliotecas, opciones de funcionamiento (letras), precedidas por un guión y seguidas por un carácter alfanumérico.

Los archivos de configuración se componen de parámetros a los que se asigna un valor entero o flotante, cadena o booleano. La forma de hacerlo es siguiendo la notación convencional de un lenguaje de programación (Parámetro = Valor).

Sirva de ejemplo el siguiente archivo de configuración, el cual, por el momento, no es necesario comprender:

Parámetros de configuración.

```
SOURCEFORMAT        = WAV
TARGETKIND          = MFCC_0_D_A
TARGETRATE          = 100000.0
SAVECOMPRESSED      = T
SAVEWITHCRC         = T
```

```
WINDOWSIZE          = 300000.0
USEHAMMING          = T
PREEMCOEF           = 0.97
NUMCHANS            = 26
CEPLIFTER           = 22
NUMCEPS             = 12
ENORMALISE          = F
```

El desarrollo completo de una aplicación se realiza en cuatro etapas: preparación de las muestras, entrenamiento, pruebas y análisis (resultados).

De forma introductoria se presentan las cuatro etapas, a fin de conocer las características de cada una, para dar paso a las siguientes dos secciones que corresponden al desarrollo de aplicaciones con HTK. A su vez, se indican las herramientas empleadas en cada fase.

B.2.1 Preparación de las muestras

A fin de construir un conjunto de HMMs, se necesita un banco de archivos de datos de voz y sus transcripciones asociadas. Antes de poder ser empleadas “en el entrenamiento deben ser convertidas a la forma paramétrica apropiada y cualquier transcripción asociada debe ser convertida para tener el formato correcto y usar las etiquetas de fonema y/o palabra” (Aparicio, 2011, p.11). Si la voz necesita ser grabada, HSLab puede ser empleada para ambas funciones, grabar y etiquetar manualmente con cualquier transcripción necesaria.

Aunque con HTK es posible parametrizar formas de onda sobre la marcha, es mejor hacerlo sólo una vez. HCopy se emplea para este propósito. Como se sugiere con el nombre, esta herramienta se usa para copiar uno o más archivos fuente a un archivo de salida. Normalmente dicha herramienta se emplea para copiar el archivo en totalidad, pero es posible emplearlo para extraer segmentos de archivos y archivos concatenados.

Configurando apropiadamente las variables, es factible convertir a formas paramétricas los archivos de entrada conforme sean leídos. Así, simplemente copiando cada archivo se obtiene la codificación requerida. La herramienta HList puede ser empleada para revisar los contenidos de cada archivo de voz y, ya que puede también convertirse sobre la marcha, puede ser usado para revisar los resultados de la conversión antes de procesar grandes cantidades de datos.

Las transcripciones también se necesitarán preparar. Típicamente las etiquetas usadas en la fuente original de transcripciones no serán exactamente como se requieren, por ejemplo, las diferencias debidas al conjunto de fonemas usado.

También, el entrenamiento de HMM podría requerir que las etiquetas sean dependientes del contexto. La herramienta HLEd es un editor de etiquetas diseñado para realizar las transformaciones requeridas a los archivos de etiqueta. La misma herramienta puede también sacar archivos a un archivo maestro de salida (MLF), el cual es usualmente más conveniente para procesamientos subsecuentes. Finalmente, en la preparación de los datos, HLStats puede reunir y mostrar estadísticas en los archivos de etiqueta y donde se requiera, HQuant puede ser usado para construir un libro de código VQ en preparación para construir un sistema discreto de probabilidad HMM.

B.2.2 Entrenamiento

El segundo paso de la construcción del sistema es la definición de la topología requerida para cada HMM escribiendo un prototipo de definición. HTK permite que los HMMs sean construidos con cualquier topología deseada. La definición de los HMM puede ser almacenada externamente como archivo simple de texto y, por lo tanto, es posible modificarlo mediante cualquier editor de texto.

Alternativamente, la distribución estándar de HTK incluye un número de ejemplos de prototipos de HMM y un script para generar las topologías más comunes automáticamente. Con excepción de las probabilidades de transición, todos los parámetros de HMM dados en el prototipo de definición son ignorados.

“El propósito del prototipo de definición es únicamente especificar el total de características y topología del HMM” (Aparicio, 2011, p.12). Los parámetros actuales serán calculados después por las herramientas de entrenamiento. Valores sensibles para las probabilidades de transición deben ser dados pero el proceso de entrenamiento es muy insensible a éstos.

Primeramente, un conjunto inicial de modelos debe ser creado. Si hay algunos datos de voz disponibles para las cuales la ubicación de las fronteras de las etiquetas están bien definidas, entonces éstos pueden usarse como datos de inicio. En este caso, las herramientas HInit y HRest proveen entrenamiento del tipo “palabra aislada” usando el conjunto de datos de entrada completa. Cada uno de los HMMs requeridos se genera individualmente. HInit lee en todos los datos de entrenamiento de inicio y recorta todos los ejemplos de los sonidos requeridos. Entonces calcula iterativamente un conjunto inicial de parámetros usando un procedimiento de segmentar k-partes. En el primer ciclo, los datos de entrenamiento son uniformemente segmentados, cada modelo de estado se relaciona con los correspondientes segmentos de datos y sus medias y varianzas son estimadas. Si los modelos de mezcla Gaussiana son entrenados, entonces una forma modificada de agrupación de k-media es usada. En el segundo ciclo y

sucesivos, la segmentación uniforme se reemplaza por alineación Viterbi. Los valores iniciales de los parámetros calculados por HInit son entonces restimados por HRest. Nuevamente, el conjunto de datos totalmente etiquetado se usa pero esta vez el procedimiento de medias es remplazado por el de restimación Baum-Welch. Cuando no hay datos de inicio disponibles, puede usarse una llamada “flat start". En este caso todos los modelos de sonidos son inicializados para ser idénticos y tener medias de estados y varianzas iguales a la media y varianza global de la voz. La herramienta HCompV puede ser usada con este propósito.

Una vez que el conjunto inicial de modelos ha sido creado, la herramienta HERest es usada para proporcionar el entrenamiento embebido usando el conjunto completo de entrenamiento. HERest realiza una restimación única Bam-Welch del conjunto completo de modelos de sonidos HMM simultáneamente. Para cada pronunciación de referencia, los modelos de sonido correspondientes son concatenados y el algoritmo de avance-regreso es usado para acumular las estadísticas del estado de ocupación, medias, varianzas, etc., para cada HMM en la secuencia.

Cuando todos los datos de entrenamiento han sido procesados, las estadísticas acumuladas son usadas para calcular las restimaciones de los parámetros de los HMM. HERest es el núcleo de las herramientas de entrenamiento de HTK. Está diseñada para procesar bases de datos extensas, provee facilidades para reducir los cálculos y puede ser ejecutado en paralelo a lo largo de la red de computadoras.

La filosofía de construcción de un sistema en HTK es que los HMMs deben ser refinados incrementalmente. Así, una progresión típica es comenzar con un conjunto simple de modelos de sonido Gaussianos independientes del contexto y entonces iterativamente refinarlos por expansión para incluir dependencia del contexto y usar múltiples mezclas de componentes de distribuciones de Gauss. Para mejorar el rendimiento para parlantes específicos las herramientas HERest y HVite pueden ser usados para adaptar a los HMMs para modelar mejor las características de parlantes particulares usando una cantidad pequeña de entrenamiento o datos de adaptación. El resultado final de esto es un sistema adaptado al hablante.

El único gran problema en la construcción de un sistema dependiente del contexto siempre es la insuficiencia de datos.

Lo más complejo, el modelo del conjunto, la mayor cantidad de datos se necesita para hacer estimaciones robustas de sus parámetros, y debido a que los datos

son usualmente limitados, un balance debe ser realizado entre la complejidad y los datos disponibles.

B.2.3 Pruebas (Reconocimiento)

HTK provee de una herramienta de reconocimiento individual llamado HVite, la cual usa el algoritmo token passing a fin de realizar el reconocimiento del habla basado en el algoritmo de Viterbi. Esta herramienta toma como entrada a una red que describe las secuencias de palabras permitidas, un diccionario que define cómo cada palabra es pronunciada y un conjunto de HMMs. Opera convirtiendo la red de palabras a una red de fonemas y entonces adjuntar la definición de HMM apropiada para cada instancia de fonema. El reconocimiento puede entonces ser realizado en cualquier de una lista de archivos de habla almacenados o directamente de la entrada de audio. HVite soporta también trifonemas de palabras cruzadas (cross-word triphones) y puede funcionar con múltiples tokens para generar conjuntos parcialmente ordenados (lattices) que contengan múltiples hipótesis. Puede también ser configurada para reestimar (rescore) lattices y conseguir alineamientos forzados.

La red de palabras necesaria para manejar HVite es usualmente cualquier ciclo de palabra simple en el cual cualquiera de ellas puede seguir a cualquiera otra palabra; también pueden ser grafos dirigidos que representen una tarea con gramática de estado finito.

Como una alternativa para especificar directamente una red de palabra, puede emplearse una notación de gramática de mayor nivel. Esta notación está basada en la forma extendida de Backus Naur (EBNF), usada en la especificación de compiladores y que es compatible con la gramática de especificación de lenguaje usada en versiones recientes de HTK. La herramienta HParse es suministrada para convertir esta notación a la red de palabras equivalente.

La herramienta HSGen toma como entrada una red y entonces atraviesa aleatoriamente la red sacando cadenas de palabras. Dichas cadenas pueden entonces ser revisadas para asegurar que corresponden a lo que se requiere. HSGen puede también calcular la perplejidad empírica de la tarea.

Finalmente, la construcción de diccionarios grandes puede envolver la fusión de gran cantidad de fuentes y la realización de una variedad de transformaciones en cada una de las fuentes. La herramienta de gestión del diccionario es HDMan suministrada para asistir a este proceso.

B.2.4 Análisis

Una vez que el reconocedor basado en HMM ha sido construido, es necesario evaluar su comportamiento. Esto es usualmente se hace transcribiendo algunas oraciones de prueba pregrabadas y relacionar la salida del reconocedor con la transcripción de las referencias correctas. Esta comparación se realiza con una herramienta llamada HResults, la cual usa programación dinámica para alinear las dos transcripciones y entonces contar las sustituciones, borrados y errores de inserción. “Las opciones son dadas para asegurar que el algoritmo y los formatos de salida usados por HResults son compatibles con aquellos usados por el instituto nacional de estándares y tecnología (NIST) de Estados Unidos” (Aparicio, 2011, p.18).

APÉNDICE C

MUSICAL INSTRUMENTS DIGITAL INTERFACE

C.1 Significado e MIDI

MIDI no forma parte de los formatos de audio digital por que no almacena muestras de sonido, no es un lenguaje musical ni describe indirectamente los sonidos; solo puede almacenar una representación del sonido, ya que indica el instrumento, la forma de tocarlo, pero el sonido finalmente depende del reproductor MIDI que lo ejecute.

Es en realidad un protocolo digital de comunicaciones, surgido del entendimiento entre fabricantes de equipos musicales electrónicos, que permitió que estos instrumentos se comunicaran entre ellos y que, por extensión, se comunicaran con las computadoras" (Serdi, 1997, p.72).

El protocolo MIDI nace en 1983, de la necesidad de conectar dos instrumentos para tocar notas simultáneamente, así como sonidos más ricos, una comisión de fabricantes japoneses y norteamericanos de instrumentos electrónicos se encargó de definir el protocolo standard que iba a permitir la conexión entre estos aparatos.

C.1.1 Ventajas del protocolo MIDI

- ✓ Permite a las computadoras utilizar herramientas de estudios de grabación y producción musical.
- ✓ Ayudan a la composición, aprendizaje y a la educación musical o la impresión de partituras.
- ✓ Ha permitido nuevos enfoques de creación de estilos musicales.
- ✓ Los archivos MIDI tienen un tamaño menor a los archivos de audio digital (wav, mp3, ffit, etc.)
- ✓ Permite una facilidad mayor para editar y modificar la música.

C.1.2 Conceptos generales

La siguiente lista de conceptos fueron sugeridos por Serdi (1997, p. 77):

- ✓ Un mensaje MIDI indica a un dispositivo una acción a ejecutar.
- ✓ Todo dispositivo que cumple la normativa MIDI dispone de una interfaz capaz de recibir y /o enviar mensajes MIDI.
- ✓ Este interfaz puede tener tres puertos diferentes: MIDI IN, MIDI OUT, MIDI THRU.
- ✓ Todo instrumento receptor debe disponer de un MIDI IN.
- ✓ Todo instrumento emisor debe disponer de un MIDI OUT.
- ✓ El MIDI THRU genera una réplica del MIDI IN, que permite encadenar varios dispositivos MIDI.

- ✓ Un secuenciador es un dispositivo capaz de grabar y reproducir mensajes MIDI.
- ✓ Una computadora equipada con el software y el hardware necesarios, puede funcionar como secuenciador.
- ✓ Aunque la conexión de varios dispositivos MIDI, puede resultar complicada, en la práctica, si nuestro sistema se compone únicamente de una computadora con una tarjeta de sonido interna y un teclado, la configuración se simplifica notablemente.

C.2 Canales MIDI

"El protocolo MIDI permite que los mensajes se envíen de dieciséis canales diferentes. Estos canales no corresponden a conexiones físicas separadas, ya que comparten un único cable, sino a direcciones lógicas" (Serdi, 1997, p.79).

Cada canal puede considerarse como un instrumento virtual.

C.3 Notas musicales en MIDI

Las notas musicales se representan en MIDI mediante un número entero positivo comprendido entre 0 y 127 (requiere de 7 bits). Con este rango cubre 11 octavas. La nota con su correspondiente octava se dan en la tabla C.1

Octava	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
-2	0	1	2	3	4	5	6	7	8	9	10	11
-1	12	13	14	15	16	17	18	19	20	21	22	23
0	24	25	26	27	28	29	30	31	32	33	34	35
1	36	37	38	39	40	41	42	43	44	45	46	47
2	48	49	50	51	52	53	54	55	56	57	58	59
3	60	61	62	63	64	65	66	67	68	69	70	71
4	72	73	74	75	76	77	78	79	80	81	82	83
5	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107
7	108	109	110	111	112	113	114	115	116	117	118	119
8	120	121	122	123	124	125	126	127				

Tabla C.1 Representación MIDI de las notas musicales.

C.4 Midinote

Midinote fue creado por Guenter Nagler en 1997, con el propósito de convertir un archivo MIDI a un archivo de texto fácilmente manipulable.

Midinote fue diseñado para trabajar con archivos MIDI compatibles al 100% con el general estándar.

No requiere instalación y se ejecuta mediante línea de comandos. Los parámetros que utiliza son:

```
> midinote [-noheader][-values][-evenlist][-units][-time]
input.mid [output.txt]
```

```
-noheader: no muestra la cabecera que explica las columnas
-values: muestra los valores crípticos para importarlos más
fácilmente a otros programas
-evenlist: muestra solo eventos listados en forma de
argumentos
-units: muestra la columna con unidades
Input.mid: archivo MIDI a ser convertido
Output.txt: Archivo resultante, si no se especifica la salida
la muestra en pantalla
```

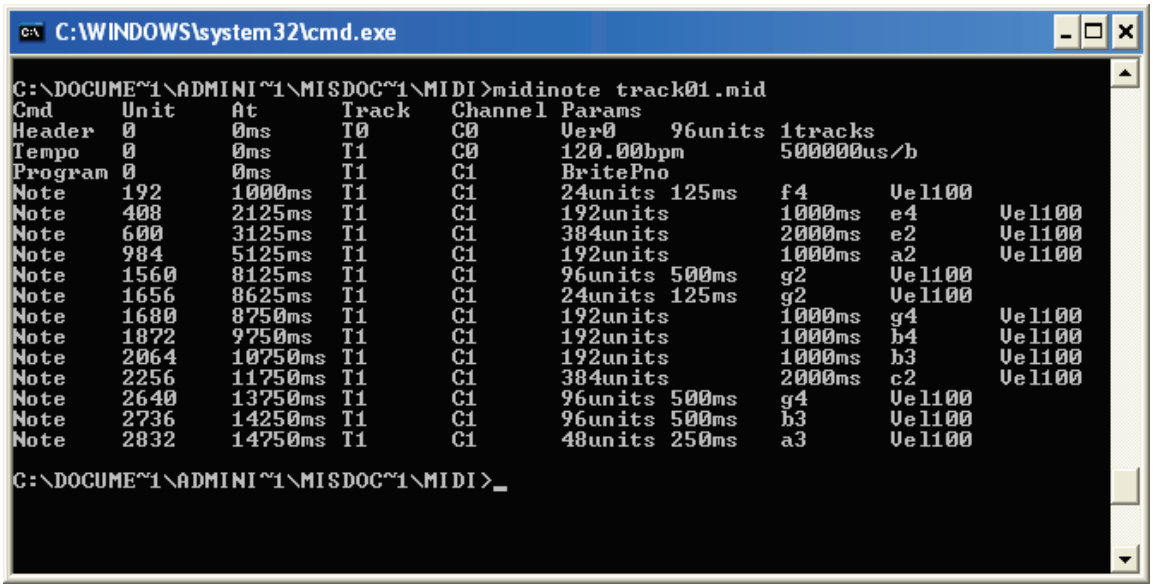


Fig. C.1 Ejemplo de la ejecución de MIDINOTE.

La figura C.1 es un ejemplo de la salida tras ejecutar MIDINOTE con una pista de muestra. Para los programas que solo requieren analizar notas, este convertidor

es muy útil, porque solo exporta la información relevante para cada nota ordenadas por tiempo, en forma de tabla usando texto simple.
Los primeros cuatro valores y el carácter del nombre del comando siempre están disponible para cada línea, y son los mismos, y están separados por un tabulador:

i. Unit, marca el inicio de un evento en unidades de tiempo MIDI.
ii. At, maraca el inicio de una nota en milisegundos
iii. Track, es el número de la pista, y comienza en 1. Cada nota es asignada a una pista y a un canal. Las pistas son pasrtes lógicas de una canción. Todas las pistas se tocan en paralelo.
iv. Channel, entre (1y 16), cuando algunos eventos no tienen canal se mandan al cananl 0. El canal 10 se reserva para los tambores.

Los comandos además añaden ciertas columnas:
Header (H):

i. Version: 0 = una pista contiene todos los eventos; 1 = más pistas con un canal para cada pista; 2 = cada pista contiene una canción
ii. Resolution: unidades por compás
iii. Trackcount: número de pistas

Tempo (T):

i. BeatsperMinute: Número de compases por minuto, inicialmente 120bm.
ii. MicrosecondsPerBeat: 500000

Note (N):

i. Lenght: Unidades de la nota
ii. Time: Duración de la nota en milisegundos
iii. Note: Valor de la nota (0-127)
iv. Velocity: Corresponde a la medida de presión ejercida cuando la nota es tocada

Program (P):

i. Program: corresponde al instrumento(0 – 127)

C.5 Note2mid

Note2mid fue creado en 1998 por Guenter Nagler.

El programa midinote convierte un archivo MIDI a un archivo de texto. Este programa reconvierte el archivo de texto a un archivo MIDI.

El programa no requiere instalación y se ejecuta mediante línea de comandos.

Para poder usar un archivo generado desde midinote, se tiene que especificar la opción –values en midinote, para que pueda ser devuelto a un MIDI. La figura C.2 ejemplifica la forma de usar midinote para poder realizar la conversión de texto a midi.

```
> midinote file.mid –values file.txt
```

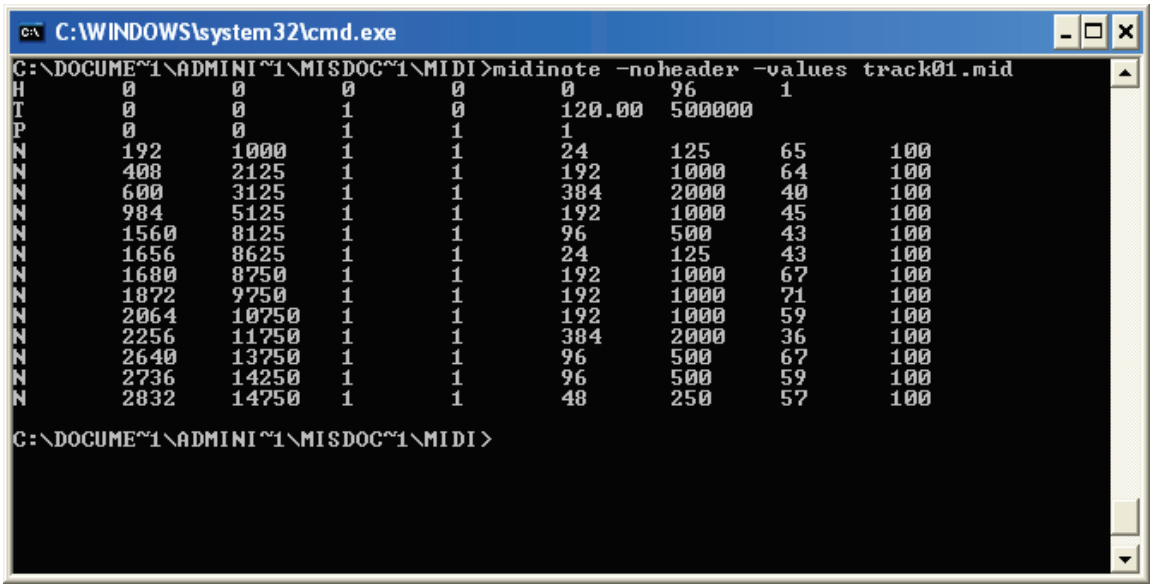


Fig. C.2 Ejemplo de la ejecucion de note2mid para la obtencion del esquema en texto de un MIDI.

```
> note2mid file.txt file.mid
```

APÉNDICE D

PROGRAMAS

D.1 Clase notas

El siguiente programa contiene una clase simple utilizada para los programas de generación de notas. Esta clase crea objetos con la siguiente información: nombre código MIDI, duración y frecuencia de aparición.

```
public class Notas{
	//Atributos
	String nombre;
	int nota;
	double duracion;
	int frec_aparicion;

	// Constructor
	public Notas(String nombre, int nota, double duracion)
	{
		this.nombre = nombre;
		this.nota = nota;
		this.duracion = duracion;
		frec_aparicion = 0;
	}
	//Metodos
	void inc_frec()
	{
		frec_aparicion++;
	}

	String getNombre()
   {
	return nombre;
   }

	int getNota()
   {
	return nota;
   }

	double getDuracion()
   {
	return duracion;
   }

	int getFrec()
   {
	return frec_aparicion;
   }
}
```

D.2 Generador de notas aleatorias con duración fija

El presente programa tiene la finalidad de generar 500 archivos de texto plano con una estructura equivalente al resultado de la ejecución de midinote.

Está compuesto por un conjunto de notas que cumplen las condiciones expuestas en el capítulo 4.

```
import java.io.*;
import java.util.*;

public class CreaMIDI_NdF
{
        //Variables generales de cabecera del archivo MIDI
        static final int CHANNEL = 1;
        static final int TRACK = 1;
        static final int UNITS = 96;
        static final int VERSION = 0;
        //Variables generales del tiempo deL MIDI
        static final int PARAM_VEL = 120;
        static final int PARAM_MEM = 500000;
        static final int VELOCITY = 100;

        public static int nota_aleatoria()
        {
              double v = Math.random();
              if (v < 0.02565)
                     return 1;
              else if (v < 0.0513)
                     return 2;
              else if (v < 0.07695)
                     return 3;
              else if (v < 0.1026)
                     return 4;
              else if (v < 0.12825)
                     return 5;
              else if (v < 0.1539)
                     return 6;
              else if (v < 0.17955)
                     return 7;
              else if (v < 0.23085)
                     return 8;
              else if (v < 0.28215)
                     return 9;
              else if (v < 0.33345)
                     return 10;
              else if (v < 0.38475)
                     return 11;
              else if (v < 0.43605)
                     return 12;
              else if (v < 0.48735)
                     return 13;
              else if (v < 0.53865)
                     return 14;
              else if (v < 0.58995)
                     return 15;
              else if (v < 0.64125)
                     return 16;
              else if (v < 0.69255)
```

```
                return 17;
        else if (v < 0.74385)
                return 18;
        else if (v < 0.79515)
                return 19;
        else if (v < 0.84645)
                return 20;
        else if (v < 0.89775)
                return 21;
        else
                return 0;
    }

public static void main(String[] args)
{
            //

            //Marcadores de avance de MIDI
            int tiempo = 0; //Indica el tiempo especifico en el que inica una nota
            int unidades_totales= 0;
            int instrumento[] = {0, 1, 7, 40, 73 };
            int val;
            int c_p1 = 0;
            int c_p2 = 100;
            int c_g = 200;
            int c_v = 300;
            int c_f = 400;

            //Partes del archivo MIDI
            String header;
            String tempo;
            String program;
            String note;
            String nombre_archivo;


            // Variables ara el manejo y control de los archivos
            //MIDI
            FileWriter MIDI[];
            MIDI = new FileWriter[500];
            PrintWriter pw[];
            pw = new PrintWriter[500];
            //Etiquetas
            FileWriter Lab[];
            Lab = new FileWriter[500];
            PrintWriter pwlab[];
            pwlab = new PrintWriter[500];
            //Notas
            FileWriter contador_notas = null;
            PrintWriter cn = null;

            //Arreglo de notas a utilizar
            Notas nota[];
            nota = new Notas[22];

            //Llenado de notas (escala 1, 2, 3 sin bemoles ni sostenidos)
            nota[0]= new Notas("S", 0, 0.125);
            nota[1]= new Notas("DO3", 60, 0.125);
            nota[2]= new Notas("FA3", 65, 0.125);
            nota[3]= new Notas("DO4", 72, 0.125);
            nota[4]= new Notas("FA4", 77, 0.125);
            nota[5]= new Notas("DO2", 48, 0.125);
```

```
                nota[6]= new Notas("FA3", 53, 0.125);
                nota[7]= new Notas("SI4", 83, 0.125);
                nota[8]= new Notas("RE3", 62, 0.125);
                nota[9]= new Notas("MI3", 64, 0.125);
                nota[10]= new Notas("SOL3", 67, 0.125);
                nota[11]= new Notas("LA3", 69, 0.125);
                nota[12]= new Notas("SI2", 59, 0.125);
                nota[13]= new Notas("RE4", 74, 0.125);
                nota[14]= new Notas("MI4", 76, 0.125);
                nota[15]= new Notas("SOL4", 79, 0.125);
                nota[16]= new Notas("LA4",81, 0.125);
                nota[17]= new Notas("SI3", 71, 0.125);
                nota[18]= new Notas("RE2", 50, 0.125);
                nota[19]= new Notas("MI2", 52,  0.125);
                nota[20]= new Notas("SOL2", 55, 0.125);
                nota[21]= new Notas("LA2", 57, 0.125);


                //Manejo de archivos
        try
        {
                                for(int k = 0; k < 500; k+=5)
                                {
                                        tiempo = 0;
                                        //Cabeceras de los archivos
                                header ="H\t0\t" + tiempo + "\t" + (TRACK-1) + "\t" +
(CHANNEL-1) + "\t" + VERSION + "\t"+ UNITS + "\t" + TRACK;
                                        tempo ="T\t0\t" + tiempo + "\t" + (TRACK) + "\t" +
(CHANNEL-1) + "\t" + PARAM_VEL + "\t"+ PARAM_MEM;
                                        program ="P\t0\t" + tiempo + "\t" + (TRACK) + "\t" +
(CHANNEL) + "\t";

                                        //Archivos para los distintos instrumentos
                                        if(c_p1 < 10)
                                        {
                                                MIDI[k+0] = new FileWriter("track00"+ c_p1 +
".txt");
                                                Lab[k+0] = new FileWriter("track00"+ c_p1 +
".lab");
                                        }
                                        else if(c_p1 < 100)
                                        {
                                                MIDI[k+0] = new FileWriter("track0"+ c_p1 +
".txt");
                                                Lab[k+0] = new FileWriter("track0"+ c_p1 +
".lab");
                                        }
                                        else
                                        {
                                                MIDI[k+0] = new FileWriter("track000.txt");
                                                Lab[k+0] = new FileWriter("track000.lab");
                                        }
                                        pw[k+0] = new PrintWriter(MIDI[k+0]);
                                        pwlab[k+0] = new PrintWriter(Lab[k+0]);
                                        c_p1++;


                                        MIDI[k+1] = new FileWriter("track"+ c_p2 + ".txt");
                                        Lab[k+1] = new FileWriter("track"+ c_p2 + ".lab");
                                        pw[k+1] = new PrintWriter(MIDI[k+1]);
                                        pwlab[k+1] = new PrintWriter(Lab[k+1]);
                                        c_p2++;
```

```
                                        MIDI[k+2] = new FileWriter("track"+ c_g + ".txt");
                                        Lab[k+2] = new FileWriter("track"+ c_g + ".lab");
                                        pw[k+2] = new PrintWriter(MIDI[k+2]);

                                        pwlab[k+2] = new PrintWriter(Lab[k+2]);
                                        c_g++;

                                        MIDI[k+3] = new FileWriter("track"+ c_v + ".txt");
                                        Lab[k+3] = new FileWriter("track"+ c_v + ".lab");
                                        pw[k+3] = new PrintWriter(MIDI[k+3]);
                                        pwlab[k+3] = new PrintWriter(Lab[k+3]);
                                        c_v++;

                                        MIDI[k+4] = new FileWriter("track"+ c_f + ".txt");
                                        Lab[k+4] = new FileWriter("track"+ c_f + ".lab");
                                        pw[k+4] = new PrintWriter(MIDI[k+4]);
                                        pwlab[k+4] = new PrintWriter(Lab[k+4]);
                                        c_f++;


                                        //Escribimos la cabecera de cada archivo
                                        for( int i = 0; i < 5; i++)
                                        {
                                                pw[k+i].println(header);
                                                pw[k+i].println(tempo);
                                                pw[k+i].println(program + instrumento[i]);
                                        }

                                        //Generacion de notas aleatorias y escritura en cada
archivo
                                        unidades_totales = 0;
                                for (tiempo= 0; tiempo < 30000; tiempo+= 125)
                                        {
                                                val = nota_aleatoria();
                                                nota[val].inc_frec();
                                                for( int i = 0; i < 5; i++)
                                                        pwlab[k+i].println(tiempo + " " +(tiempo
+ (int)(nota[val].getDuracion()*1000)) +" "+ nota[val].getNombre() );
                                                if( val != 0)
                                                {
                                                note ="N\t"+ unidades_totales+ "\t"+ tiempo+
"\t"+ TRACK +"\t"+ CHANNEL+ "\t"+ (UNITS/4) +"\t" + (int)(nota[val].getDuracion()*1000)+
"\t"+ nota[val].getNota() + "\t" + VELOCITY;
                                        for( int i = 0; i < 5; i++)
                                                        pw[k+i].println(note);
                                                }
                                                unidades_totales+=(UNITS/4);
                                        }
                                }

                                //Codigo que regresa el total de cada  nota
                                contador_notas = new FileWriter("notas.txt");
                        cn = new PrintWriter(contador_notas);
                                for (int j = 0; j < 22; j++)
                                        cn.println(nota[j].getNota()+ "\t" +
nota[j].getFrec());

                        }
                        catch (Exception e) {
                    e.printStackTrace();
                }
```

```
                finally
                {
            try
                        {
            // Nuevamente aprovechamos el finally para
            // asegurarnos que se cierra el fichero.
                        for( int i = 0; i < 500; i ++)
                        {
                    if (null != MIDI[i])
                    MIDI[i].close();
                                    if (null != Lab[i])
                    Lab[i].close();
                        }

                        if (null != contador_notas)
               contador_notas.close();
            }
                        catch (Exception e2)
                        {
               e2.printStackTrace();
            }
        }
    }
}
```

D.3 Generador de notas aleatorias de duración variable

Este programa es similar al anterior, la diferencia está en que la duración de cada nota también se elige de manera aleatoria con cierta probabilidad.

```
import java.io.*;
import java.util.*;

public class CreaMIDI_NdV
{
	//Variables generales de cabecera del archivo MIDI
	static final int CHANNEL = 1;
	static final int TRACK = 1;
	static final int UNITS = 24;
	static final int VERSION = 0;
	//Variables generales del tiempo deL MIDI
	static final int PARAM_VEL = 120;
	static final int PARAM_MEM = 500000;
	static final int VELOCITY = 100;
	//Contadores de la  duracion
	static int semicorchea = 0;
	static int corchea = 0;
	static int negra = 0;
	static int blanca = 0;
	static int redonda = 0;

	//Metodo que escoge la duracion de la nota con cierta probabilidad
	public static double figura_variable()
	{
		double v = Math.random();
		if ( v < 0.1)
		{
			semicorchea++;
			return 1;
		}
		else if ( v < 0.28)
		{
			corchea++;
			return 2;
		}
		else if ( v < 0.52)
		{
			negra++;
			return 4;
		}
		else if ( v < 0.81)
		{
			blanca++;
			return 8;
		}
		else
		{
			redonda++;
			return 16;
		}
	}

	//Metodo que devuelve la nota con cierta probabilidad
	public static int nota_aleatoria()
	{
```

```
            double v = Math.random();
            if (v < 0.02565)
                    return 1;
            else if (v < 0.0513)
                    return 2;
            else if (v < 0.07695)
                    return 3;
            else if (v < 0.1026)
                    return 4;
            else if (v < 0.12825)
                    return 5;
            else if (v < 0.1539)
                    return 6;
            else if (v < 0.17955)
                    return 7;
            else if (v < 0.23085)
                    return 8;
            else if (v < 0.28215)
                    return 9;
            else if (v < 0.33345)
                    return 10;
            else if (v < 0.38475)
                    return 11;
            else if (v < 0.43605)
                    return 12;
            else if (v < 0.48735)
                    return 13;
            else if (v < 0.53865)
                    return 14;
            else if (v < 0.58995)
                    return 15;
            else if (v < 0.64125)
                    return 16;
            else if (v < 0.69255)
                    return 17;
            else if (v < 0.74385)
                    return 18;
            else if (v < 0.79515)
                    return 19;
            else if (v < 0.84645)
                    return 20;
            else if (v < 0.89775)
                    return 21;
            else
                    return 0;
    }

public static void main(String[] args)
{
             //
             //Marcadores de avance de MIDI
             int tiempo = 0; //Indica el tiempo especifico en el que inica una nota
             int unidades_totales= 0;
             int instrumento[] = {0, 1, 7, 40, 73 };
             int val;
             double fig;
             int c_p1 = 0;
             int c_p2 = 100;
             int c_g = 200;
             int c_v = 300;
             int c_f = 400;
```

```
                //Partes del archivo MIDI
                String header;
                String tempo;
                String program;
                String note;
                String nombre_archivo;


                // Variables ara el manejo y control de los archivos
                //MIDI
                FileWriter MIDI[];
                MIDI = new FileWriter[500];
                PrintWriter pw[];
                pw = new PrintWriter[500];
                //Etiquetas
                FileWriter Lab[];
                Lab = new FileWriter[500];
                PrintWriter pwlab[];
                pwlab = new PrintWriter[500];
                //Notas
                FileWriter contador_notas = null;
                PrintWriter cn = null;

                //Arreglo de notas a utilizar
                Notas nota[];
                nota = new Notas[22];

                //Llenado de notas (escala 1, 2, 3 sin bemoles ni sostenidos)
                nota[0]= new Notas("S", 0, 0.125);
                nota[1]= new Notas("DO3", 60, 0.125);
                nota[2]= new Notas("FA3", 65, 0.125);
                nota[3]= new Notas("DO4", 72, 0.125);
                nota[4]= new Notas("FA4", 77, 0.125);
                nota[5]= new Notas("DO2", 48, 0.125);
                nota[6]= new Notas("FA2", 53, 0.125);
                nota[7]= new Notas("SI4", 83, 0.125);
                nota[8]= new Notas("RE3", 62, 0.125);
                nota[9]= new Notas("MI3", 64, 0.125);
                nota[10]= new Notas("SOL3", 67, 0.125);
                nota[11]= new Notas("LA3", 69, 0.125);
                nota[12]= new Notas("SI2", 59, 0.125);
                nota[13]= new Notas("RE4", 74, 0.125);
                nota[14]= new Notas("MI4", 76, 0.125);
                nota[15]= new Notas("SOL4", 79, 0.125);
                nota[16]= new Notas("LA4",81, 0.125);
                nota[17]= new Notas("SI3", 71, 0.125);
                nota[18]= new Notas("RE2", 50, 0.125);
                nota[19]= new Notas("MI2", 52,  0.125);
                nota[20]= new Notas("SOL2", 55, 0.125);
                nota[21]= new Notas("LA2", 57, 0.125);

                //Manejo de archivos
        try
        {
                                for(int k = 0; k < 500; k+=5)
                                {
                                        tiempo = 0;
                                        //Cabeceras de los archivos
                                header  ="H\t0\t"  +  tiempo  +  "\t"  +  (TRACK-1)  +  "\t"  +
(CHANNEL-1) + "\t" + VERSION + "\t"+ (UNITS*4) + "\t" + TRACK;
                                        tempo  ="T\t0\t"  +  tiempo  +  "\t"  +  (TRACK)  +  "\t"  +
(CHANNEL-1) + "\t" + PARAM_VEL + "\t"+ PARAM_MEM;
```

```
program ="P\t0\t" + tiempo + "\t" + (TRACK) + "\t" +
(CHANNEL) + "\t";

//Archivos para los distintos instrumentos
if(c_p1 < 10)
{
       MIDI[k+0] = new FileWriter("track00"+ c_p1 +
".txt");
       Lab[k+0] = new FileWriter("track00"+ c_p1 +
".lab");
}
else if(c_p1 < 100)
{
       MIDI[k+0] = new FileWriter("track0"+ c_p1 +
".txt");
       Lab[k+0] = new FileWriter("track0"+ c_p1 +
".lab");
}
else
{
       MIDI[k+0] = new FileWriter("track000.txt");
       Lab[k+0] = new FileWriter("track000.lab");
}
pw[k+0] = new PrintWriter(MIDI[k+0]);
pwlab[k+0] = new PrintWriter(Lab[k+0]);
c_p1++;


MIDI[k+1] = new FileWriter("track"+ c_p2 + ".txt");
Lab[k+1] = new FileWriter("track"+ c_p2 + ".lab");
pw[k+1] = new PrintWriter(MIDI[k+1]);
pwlab[k+1] = new PrintWriter(Lab[k+1]);
c_p2++;

MIDI[k+2] = new FileWriter("track"+ c_g + ".txt");
Lab[k+2] = new FileWriter("track"+ c_g + ".lab");
pw[k+2] = new PrintWriter(MIDI[k+2]);

pwlab[k+2] = new PrintWriter(Lab[k+2]);
c_g++;

MIDI[k+3] = new FileWriter("track"+ c_v + ".txt");
Lab[k+3] = new FileWriter("track"+ c_v + ".lab");
pw[k+3] = new PrintWriter(MIDI[k+3]);
pwlab[k+3] = new PrintWriter(Lab[k+3]);
c_v++;

MIDI[k+4] = new FileWriter("track"+ c_f + ".txt");
Lab[k+4] = new FileWriter("track"+ c_f + ".lab");
pw[k+4] = new PrintWriter(MIDI[k+4]);
pwlab[k+4] = new PrintWriter(Lab[k+4]);
c_f++;


//Escribimos la cabecera de cada archivo
for( int i = 0; i < 5; i++)
{
       pw[k+i].println(header);
       pw[k+i].println(tempo);
       pw[k+i].println(program + instrumento[i]);
}
```

```
                                        //Generacion de notas aleatorias y escritura en cada
archivo
                                        unidades_totales = 0;
                        for (tiempo= 0; tiempo <= 30000; tiempo += (125*fig))
                                        {
                                                fig = figura_variable();
                                                val = nota_aleatoria();
                                                nota[val].inc_frec();
                                                for( int i = 0; i < 5; i++)
                                                        pwlab[k+i].println((tiempo*10000) +" "+
((tiempo*10000)       +       (int)(nota[val].getDuracion()*10000000*fig)       )+"       "+
nota[val].getNombre());
                                                if( val != 0)
                                                {
                                                note  ="N\t"+  unidades_totales+  "\t"+  tiempo+
"\t"+     TRACK     +"\t"+     CHANNEL+     "\t"     +     (int)(UNITS*fig)     +"\t"     +
(int)(nota[val].getDuracion()*1000*fig)+ "\t"+ nota[val].getNota() + "\t" + VELOCITY;
                                for( int i = 0; i < 5; i++)
                                                        pw[k+i].println(note);
                                                }
                                                unidades_totales+=(UNITS*fig);
                                        }
                                }

                                //Codigo que regresa el total de cada  nota
                                contador_notas = new FileWriter("notas.txt");
                        cn = new PrintWriter(contador_notas);
                                for (int j = 0; j < 22; j++)
                                        cn.println(nota[j].getNota()+          "\t"          +
nota[j].getFrec());
                                cn.println("Semicorchea\t" + semicorchea);
                                cn.println("Corchea\t" + corchea);
                                cn.println("Negra\t" + negra);
                                cn.println("Blanca\t" + blanca);
                                cn.println("Redonda\t" + redonda);

                        }
                        catch (Exception e) {
                    e.printStackTrace();
                }
                        finally
                        {
                   try
                                {
                   // Nuevamente aprovechamos el finally para
                   // asegurarnos que se cierra el fichero.
                                for( int i = 0; i < 500; i ++)
                                {
                              if (null != MIDI[i])
                              MIDI[i].close();
                                        if (null != Lab[i])
                              Lab[i].close();
                                }

                                if (null != contador_notas)
                      contador_notas.close();
                   }
                                catch (Exception e2)
                                {
                      e2.printStackTrace();
                   }
                }}
```

REFERENCIAS

Aguilera P. (2007), *Reconocimiento de voz usando HTK*¸ España: Universidad de Sevilla, Departamento de la Teoría de Señal y Telecomunicaciones. Tesis de licenciatura en Ingeniería en Telecomunicaciones.

Aparicio A. (2011). *Manual de uso y desarrollo de aplicación con HTK.* México: Instituto Politécnico Nacional, Centro de Investigación en Computación.

Arribas J. (2006). *Psicoacústica.* Recuperado el 12 de Julio de 2013 de http://www.lpi.tel.uva.es/~nacho/docencia/ing_ond_1.htm.

Basogain X. (2008). *Redes Neuronales Artificiales***.** España: Escuela Superior de Ingeniería de Bilbao.

Date C. J. (2001). *Introducción a los sistemas de bases de datos.* (7ª Ed.) México: Pearson Educación.

Devijver P. R. y Kittler J. (1982), *Pattern Recognition: A Statistical Approach.* New Jersey: Prentice-Hall.

Durey A. S. (2003). *Melody Spotting Using Hidden Markov Models.* Georgia: Georgia Institute of Technology, School of Electrical and Computer Engineering. Tesis de licenciatura.

Elmasri R. y Navathe S. B. (2007). *Fundamentos de Sistemas de Bases de Datos*. (5ª Ed.) México: Pearson Educación.

Fernández M. A. (2002). Breve historia de la música. Recuperado el 20 Marzo del 2013 de http://imagenes.mailxmail.com/cursos/pdf/6/música-breve-historia-29456.pdf

Grabner H. (2001). *Teoría general de la música*. Barcelona: Akal.

Güimi (2008). Historia de la computación, Recuperado el 1 Abril de2013 de guimi.net.

Gutiérrez A. (2009). *Fundamentos de la computación*. México: Polilibros.

Instituto Politécnico de Madrid. Escuela Universitaria Técnica de Telecomunicación. (2000), Manual Técnico de sonido**.** Recuperado el 08 Julio del 2013 de http://www.diac.upm.es/escuela.

Knuth D. E. (2002). *The Art of Computer Programming: Volume 1. Fundamental Algorithms***.** (2ª Ed.) Massachusetts: Addison-Wesley Publishing Company.

Kreyszig E. (1982). *Matemáticas Avanzadas para Ingeniería. Vol. 1.* (3ª. Ed.) México: Limusa.

Logan B. (2000). *Mel Frequency Cepstral Coefficients for Music Modeling.* Recuperado el 01 de Octubre de 2013 de http://apotheca.hpl.hp.com/ftp/pub/compaq/CRL/publications/logan/.

Lopez A. (2000). *Ingeniería de Ondas: Formatos de audio digital.* España: Universidad de Valladolid.

Martínez F. J. (2009). *Tutorial Web de Técnicas de Digitalización de Audio para la asignatura Tratamiento Digital de Audio*. España: Universidad Carlos III de Madrid, Departamento de Teoría de la Señal y Comunicaciones. Tesis de licenciatura en Ingeniería Técnica en Telecomunicaciones.

Miyara F. (1999). *Acústica y Sistemas de sonido*, Argentina: UNR.

Moreau N. (2002). *HTK v.3.1 Basic Tutorial.* Berlín: Technische Universität.

Navarrete T. (2007). *Detección de anomalías en la carga de un procesador, utilizando modelos ocultos de Markov.* México: Instituto Tecnológico de Morelia. Tesis de maestría en Ciencias de la Computación.

Pelinsky, R. (2000). *Invitación a la Etnomusicología: Quince fragmentos y un tango*. España: AKAL.

Pohlmann K., (2002). *Principles of digital audio.* (5ª Ed.) New York: McGraw-Hill.

Rabiner L. (1989) *A tutorial on Hidden Markov Models and Selected Applications in speech Recognition.* Recuperado de la Base de datos IEEE 77: 257–286.

Salcedo F. J. (2005). *Modelos Ocultos de Markov: Del reconocimiento de la voz a la música.* Granada: Universidad de Granada.

San Martín J. E. (2012). *Herramientas complementarias al audio.* Recuperado el 08 de Julio de 2013 de http://www.astormastering.com.ar/Clase_2_Analisis_Espectral_y_Herramientas_Complementarias.pdf.

Scheirer D. E, (1998). *Tempo and beat analysis of acoustic musical signal*. Recuperado el 1 de Octubre de 2013 de http://www.music.mcgill.ca/~nester/analysis_beat_tempo_scheirer.pdf.

Seguí S. (1984). *Curso de solfeo.* España: Unión musical española.

Sergi J. P., (1997). *Audio digital y MIDI***.** España: Anaya.

Serrano A. J., Soria E., Martín J. D. (2009). *Redes Neuronales Artificiales***.** España: Universidad de Valencia.

Silberschatz A., Korth H. F. y Sudarshan S. (2002). *Fundamentos de Bases de Datos*. (4ª Ed.) McGraw-Hill.

Tippens P. E. (2002). *Física: conceptos y aplicaciones.* (8ª. Ed.) México: McGraw-Hill.

Torres J. L. (1972) *Educación Musical.* México: Porrúa.

Tresguerres J. A. F. (2009). *Anatomía y fisiología del cuerpo humano***.** España: Mc Graw Hill.

Young S, Kershaw D., Odell J., Ollason D., Valtchev V., Woodland P. (2000). *HTKBook.* Estados Unidos: Microsoft Corporation.