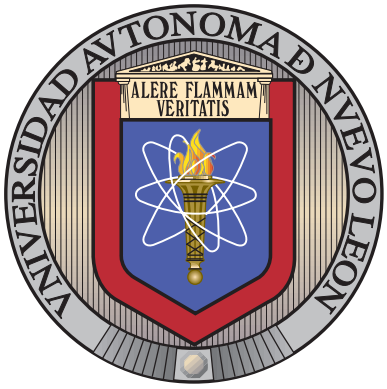
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Licenciatura



Por definir

por

Iván Alejandro Guerra López

en opción al grado de

Ingeniero en Tecnología de Software

San Nicolás de los Garza, Nuevo León agosto 2014

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Licenciatura

Por definir

por

Iván Alejandro Guerra López

en opción al grado de

Ingeniero en Tecnología de Software

San Nicolás de los Garza, Nuevo León agosto 2014

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Licenciatura

Los miembros del Comité de Tesis recomendamos que la Tesis «Por definir», realizada por el alumno Iván Alejandro Guerra López, con número de matrícula 1425688, sea aceptada para su defensa como opción al grado de Ingeniero en Tecnología de Software.

El Comité de Tesis

Dra. Satu Elisa Schaeffer

Asesor

Dr. César Guerra Torres

Revisor

Dr. Romeo Sánchez Nigenda

Revisor

Vo. Bo.

M.C. Arnulfo Treviño Cubero

División de Estudios de Licenciatura

San Nicolás de los Garza, Nuevo León, agosto 2014

Agradecimientos

Quiero agradecer a las personas que conforman la Facultad de Ingeniería Mecánica y Eléctrica y en general a la Universidad Autónoma de Nuevo León ya que de alguna manera gracias a ellos estoy aquí.

También agradezco a todos los profesores que me dieron la enseñanza necesaria y el apoyo que busqué durante mis estudios. A mis compañeros y amigos que me ayudaron a corregir mis errores y a resolver mis dudas cuando las tenía.

Agradezco profundamente a mi asesora de tesis la Dra. Satu Elisa Schaeffer por apoyarme en mi trabajo, por todo lo que me enseñó, por todas las veces que me llamó la atención, por las oportunidades que me dio y en general por formar parte de casi toda mi formación profesional.

A los miembros del comité de tesis el Dr. César Guerra Torres y el Dr. Romeo Sánchez Nigenda por sus comentarios y sugerencias en este trabajo.

Agradezco a mis padres Víctor Alejandro Briones Vázquez y Gloria Leticia Segovia Sáenz porque ellos creyeron en mi aún cuando ya no veía solución a un problema o cuando creía que ya no podía continuar, por inculcarme el estudio cuando era pequeño, por guiarme por el camino correcto y sobre todo por apoyarme y por estar ahí cuando los necesitaba.

Resumen

Iván Alejandro Guerra López.

Candidato para el grado de Ingeniero en Tecnología de Software.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

Por definir

Número de páginas: 53.

Objetivos y método de estudio: El objetivo de este trabajo es desarrollar un software que utilice técnicas de análisis de señales para reconocer ciertos patrones sonoros en la música, de esta manera generar una lista de reproducción basada en el ritmo obtenido de los archivos reproducidos. Adicionalmente se desea experimentar con efectos de transición entre pistas musicales para dar un efecto de reproducción sin pausa entre pistas que aparentemente tienen un ritmo parecido, diferenciar y detectar el principio y final de una pista, así como el momento óptimo para realizar la transición entre pistas.

El desarrollo de software consiste en su mayor parte de análisis de información

obtenida de archivos de sonido, el sistema trabaja en tiempo real, por lo que todo ocurre mientras se reproduce el sonido. La intención es conocer la opinión de los usuarios en cuanto a eficiencia.

Contribuciones y conclusiones: La contribución principal de este trabajo es la aplicación de algoritmos de reconocimiento de patrones y técnicas de análisis de señales en un software para lograr una reproducción musical sin pausas, con la intención de mejorar la experiencia musical del usuario.

El programa generado fue probado bajo ciertas circunstancias de software y hardware para obtener un análisis de funcionamiento detallado y que servirá para mejorar a futuro.

Firma del asesor: _____

Dra. Satu Elisa Schaeffer

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Desde los primeros años en que el humano fabricó y utilizó herramientas, la música ha formado parte de nuestras vidas, ya sea de forma directa o indirecta siempre habrá música. Inclusive cuando hablamos, a veces sentimos la necesidad de decir frases con cierta entonación que sigue un ritmo musical. No cabe duda que la música es importante para nosotros.

Muchos lo ven como una forma de distraerse del mundo, o como una forma de relajarse. Dependiendo de nuestro estado de ánimo buscaremos ciertos ritmos rápidos o lentos, fuertes o suaves, a veces buscando inspiración o concentración o simplemente escuchar música de pasatiempo.

1.1 Motivación

Desde un punto de vista más técnico, la música es una composición de sonidos que en conjunto siguen un ritmo específico. Por alguna razón buscamos escuchar sonidos que sigan un ritmo, pero preferimos evitar aquellos sonidos que carecen de el, algo que solemos llamar ruido.

Dependiendo del gusto de las personas, muchas consideran que ciertos géneros de música solamente son ruido, mientras que otras los defienden. Los gustos varían de

persona a persona, toman influencia en aspectos como la forma en que nos educan, las personas con quien convivimos, las costumbres de la región, etcétera. Estos gustos a veces se ven alterados con el tiempo, factores como cambiar de región o las modas también afectan los gustos.

Sin importar los géneros musicales, el ruido es algo que perturba lo que escuchamos ya sea porque la calidad de audio es mala, o por los anuncios que algún servicio incluya en su reproductor o simplemente las pausas entre pistas de audio.

Sabiendo que hay muchas variables que afectan lo que escuchamos, siempre se trata de llegar a un estado de armonía aún sabiendo que cualquier cosa puede interrumpir este estado. ¿Cómo evitar la interrupción de la sensación que produce escuchar una pista de sonido? La respuesta es mezclar las pistas de audio en una sola usando la reproducción sin pausas.

1.2 Hipótesis

Es posible lograr mezclar el inicio y final de dos pistas de audio con ritmo similar mediante el uso de algoritmos de reconocimiento de patrones de tal forma que no se perciba el cambio entre melodías para lograr una reproducción sin pausas.

1.3 Objetivos

El objetivo principal es crear una herramienta capaz de reproducir archivos de sonido de forma continua, y mediante un algoritmo mezclar el inicio y final de las pistas reproducidas de forma que conserven su ritmo y no se perciba el cambio de pista.

Los objetivos específicos son:

- Crear un reproductor de audio que haga uso de esta herramienta para crear una transición casi imperceptible entre pistas.
- Realizar pruebas con usuarios con la música de su gusto.
- Tratar de implementar la herramienta a alguna otra plataforma ya sea en una aplicación móvil o una aplicación web.

1.4 Estructura de la tesis

El presente trabajo de tesis está organizado de la siguiente manera:

En el capítulo 1 se habla en forma general sobre la intención de la tesis y se introducen algunos conceptos en la sección de hipótesis y objetivos que son utilizados durante la implementación del proyecto de tesis.

En el capítulo 2 se habla de los antecedentes a los temas que se abordan. Se definen conceptos generales y específicos a los temas *procesamiento de señales* y *reconocimiento de patrones*.

En el capítulo 3 se presentan trabajos que de alguna manera se relacionan con el tema y proyecto de tesis. Los trabajos presentados se dividen en aquellos que investigan sobre la detección rítmica en una pista de sonido y aquellos que implementan alguna herramienta o aplicación utilizando la misma.

En el capítulo 4 se define la metodología; se mencionan las herramientas utilizadas y se muestra el desarrollo del trabajo realizado en cuanto a procesamiento de señales y reconocimiento de patrones. También se muestra el diseño e implementación del software, los datos procesados y la forma en que se trabajan los datos.

En el capítulo 5 se evalúa el desempeño del software desarrollado bajo ciertos

parámetros, se muestran resultados de pruebas con usuarios y finalmente se tiene una discusión acerca de lo bueno y lo malo de la implementación y el algoritmo así como del grado de aceptación por parte de los usuarios hacia el proyecto.

En el capítulo 6 se resumen los resultados obtenidos con el proyecto. También se habla sobre las áreas de oportunidad del mismo y el trabajo a futuro.

Capítulo 2

Antecedentes

La frecuencia es la magnitud que mide la cantidad de pulsos que ocurren en un instante de tiempo. Determina la cantidad de latidos del corazón o el *tempo*[1] musical; ambos medidos en *pulsos por minuto* (bpm, del inglés *beats per minute*). Para determinar la frecuencia en una onda es necesario saber cuál es su longitud de onda. La longitud de onda se determina por la distancia entre dos *crestas*[2] o dos *valles*[3] consecutivos.

La figura 2.1 muestra un ejemplo de gráfica de una onda cuya frecuencia disminuye con el progreso del tiempo.

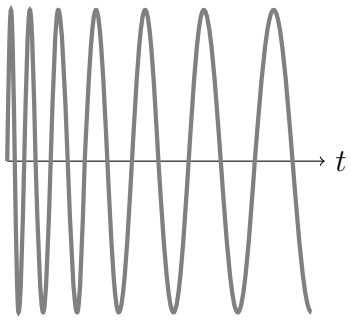


Figura 2.1 – La onda disminuye su frecuencia con la progresión de tiempo t.

Smith [19] menciona que el oído humano es un órgano sumamente complejo. Es sensible a la vibración de las ondas mecánicas producidas a cierta frecuencia. Específicamente

[1]El *tempo* es la velocidad con la que se ejecuta una pieza musical.
[2]La *cresta* es el punto más alto de una onda.
[3]El *valle* es el punto más bajo de una onda.

el umbral de audición va desde los 20Hz hasta los 20kHz. El oído es menos sensible a frecuencias más bajas o más altas a las anteriores.

La *cóclea* es una estructura interna del oído llena del líquido por el cual viajan las ondas recibidas, solo una porción de las ondas logran pasar y estimulan el nervio auditivo. De esta manera podemos oir.

Los sonidos que se reproducen en la computadora o en cualquier otro dispositivo electrónico están preparados para el oído humano. Al procesar el sonido se filtran las frecuencias no audibles, de esta manera se ahorra espacio, también se procesa el sonido a una velocidad tal que los fragmentos de la pista de sonido no produzcan silencios.

Generalmente estos fragmentos son almacenados en archivos los cuales son tratados de alguna manera para reproducirlos. Estos archivos contienen información básica que indica la forma en que se reproducirá el sonido, como el número de fragmentos de sonido en el archivo o la cantidad de fragmentos que se reproducen por segundo. El archivo también contiene los datos del sonido a reproducir comprimidos en algún formato que el reproductor puede interpretar.

Para reproducir un sonido en algún dispositivo electrónico se lleva a cabo una serie de procesos algorítmicos que sirven para interpretar los datos almacenados en los archivos, algo conocido como *decodificación*. Una vez decodificados los datos se obtiene información que describe una *onda acústica* [5].

Cuando se reproduce un sonido lo que en realidad se escucha es una representación del sonido real, que ha sido procesada. La onda que se reproduce puede ser muy similar pero no igual al sonido original. Este sonido producido es conocido como *sonido digital*.

2.1 Procesamiento de señales

La producción de sonido digital es un proceso de transformación de una señal analógica a digital. Baher [1] describe tres etapas:

Muestreo. El proceso en el que una señal analógica es transformada en *datos discretizados*.

Cuantificación. Es la *digitalización* de los datos obtenidos durante el muestreo de la señal.

Codificación. El proceso de almacenado de la señal discretizada en algún formato de *codec*.

Estas tres etapas se discuten en detalle a continuación.

2.1.1 Muestreo

Para poder trabajar con sonidos digitales, es necesario reducir las señales deben a muestras discretas de un dominio de tiempo discreto. Esta operación es llamada *muestreo*.

El muestreo consiste en tomar valores de una señal de tiempo continuo en instantes de tiempo múltiplos de T, llamado intervalo de muestreo. La cantidad $F_s = 1/T$ es llamada *frecuencia de muestreo* [17]. En la figura 2.2 se ilustra un ejemplo del proceso de muestreo.

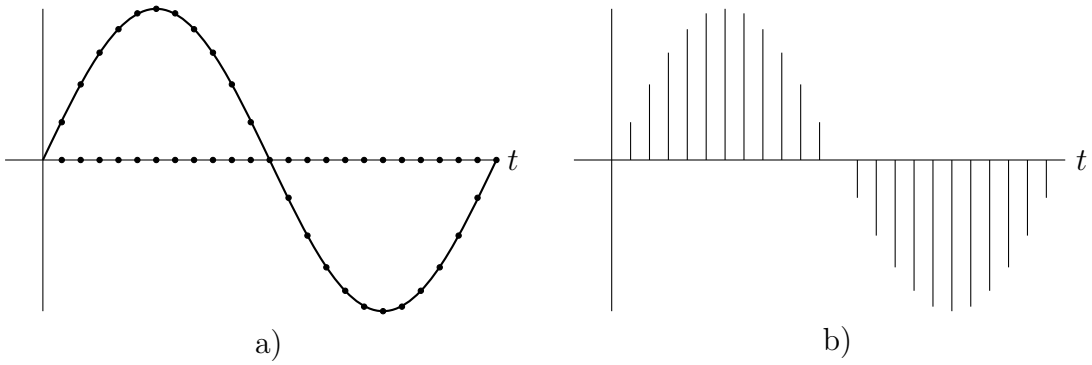


Figura 2.2 – En a) se toman muestras de una onda en intervalos de tiempo regulares t, en b) se hace una representación digital de la onda.

2.1.2 Cuantificación

Los datos obtenidos del muestreo ahora pueden ser representados como una señal de tiempo discreto y valores continuos. Con la cuantificación los valores se discretizan. El valor de cada muestra es aproximado entonces con un valor de un conjunto finito de posibles valores. La diferencia entre el valor continuo y su aproximación se le denomina error de cuantificación [1]. En la figura 2.3 se puede ver un ejemplo de cuantificación de una señal.

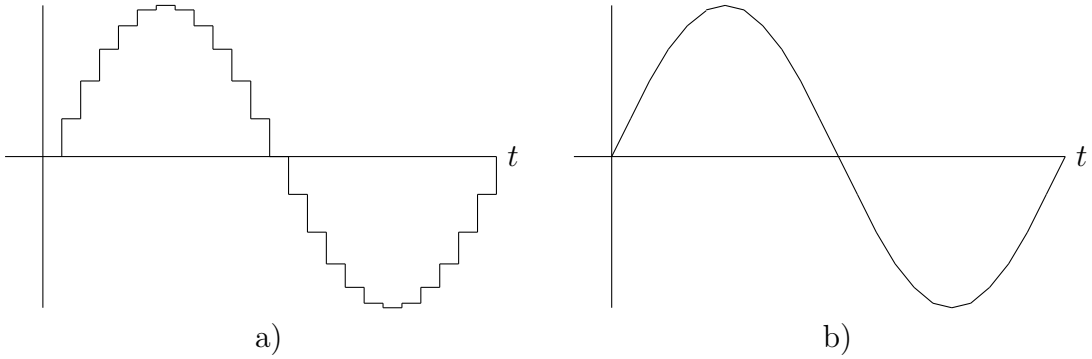


Figura 2.3 – La representación digital es discretizada en a) y posteriormente se suavizan los valores de ésta en b) para dar una forma parecida a la onda original.

2.1.3 Codificación

Después de la cuantificación se tiene una señal de timepo discreto con valores discretos como la de la figura 2.4 que se puede almacenar en un *contenedor*; este contenedor comprime la información mediante algún algoritmo de codificación, puede ser con o sin pérdida. La compresión con pérdida usa algoritmos que almacenan una señal similar pero no igual a la original. La compresión sin pérdida usa un algoritmo que almacena los datos originales, generalmente ocupan una cantidad de espacio en disco mayor [19].

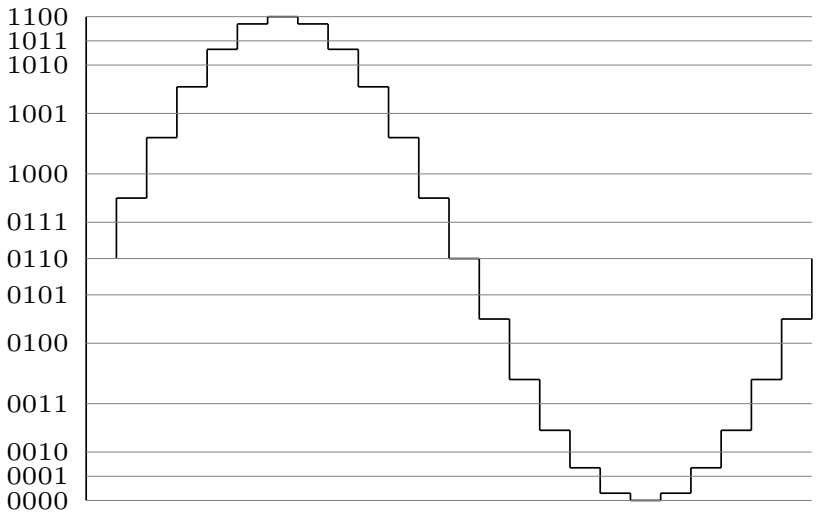


Figura 2.4 – La señal cuantificada puede representarse en forma de datos discretizados.

Independientemente del metodo de compresión, con la codificación se obtiene la información de la señal en un formato binario como se muestra en la figura 2.5.

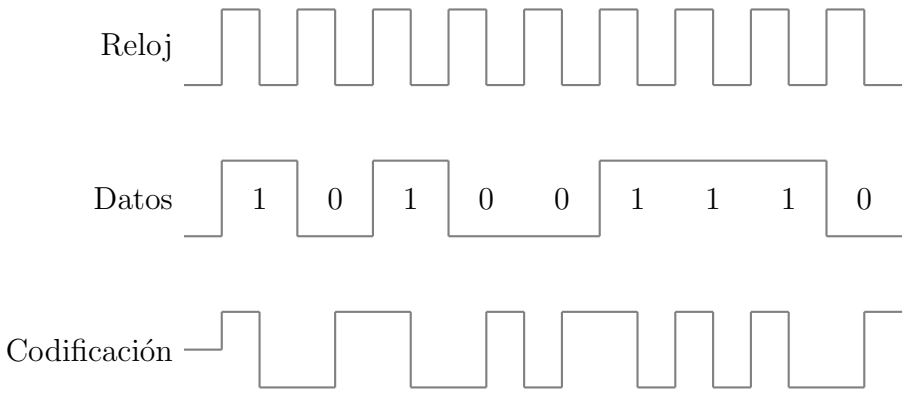


Figura 2.5 – Proceso de codificación de una señal transformada a datos.

2.2 Transformada rápida de Fourier

Durante el procesamiento de señales puede ser de utilidad analizar frecuencias por separado. La *transformada rápida de Fourier* permite separar una señal de onda en sus diferentes frecuencias. La figura 2.6 muestra un ejemplo gráfico del funcionamiento del algoritmo.

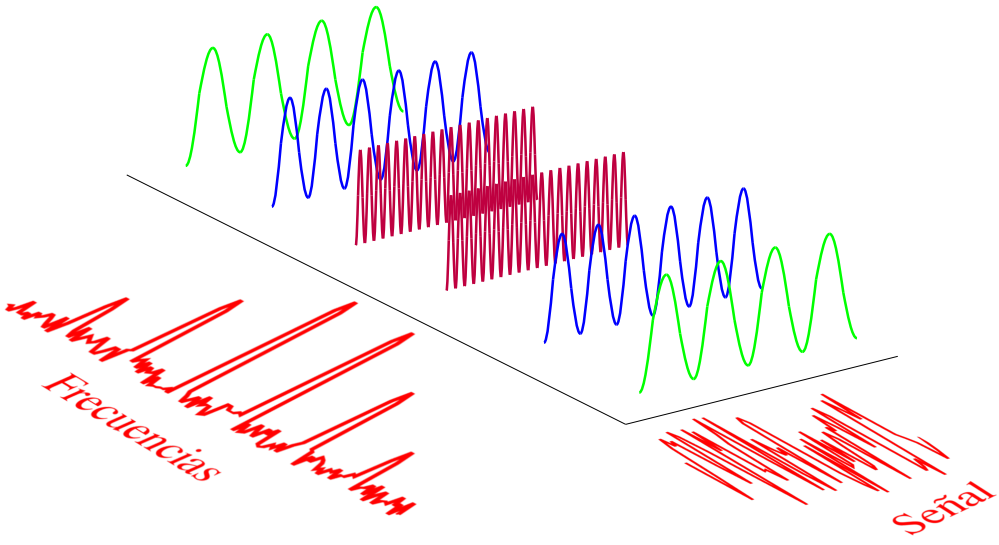


Figura 2.6 – Comparación entre una señal y sus frecuencias separadas en ondas individuales.

A continuación se describe uno de los algoritmos más usados [20]. Se asume que

$N = 2^R$ donde R es un entero positivo:

$$H_k = \sum_{j=0}^{N-1} h_j W^{jk}, \text{donde } W = e^{\pm \frac{2\pi i}{N}} \tag{2.1}$$

2.3 Reconocimiento de patrones

El reconocimiento de patrones consiste en la extracción de características importantes en imagenes, sonidos, cadenas de ADN, etcétera. Watanabe [22] menciona que un patrón es una entidad, vagamente definida, la cual puede nombrarse.

Kittler [12] en sus notas de seminario menciona que es posible modelar un sistema de tres estados basado en el sistema perceptual de un ser humano:

Adquisición de datos. Consiste en recolectar información de algún medio ya sea una cámara, un micrófono, o cualquier sensor que genere datos en bruto.

Extracción de características. Mediante el uso de algoritmos tratar de encontrar información relevante dentro de los datos adquiridos.

Toma de desiciones. Cuando se tienen las características relevantes ahora se clasifican dependiendo del propósito del programa.

Existen diferentes aproximaciones a este problema, a continuación se describen brevemente algúnas de la más utilizadas [11].

Comparación de plantillas Este es el modelo más sencillo y quizá uno de los primeros en existir. Consiste en comparar una plantilla con la información de una muestra obtenida buscando similitudes

Modelo estadístico Este modelo se basa en la teoría de probabilidad y estadística y supone que se tiene un conjunto de medidas numéricas con distribuciones de probabilidad conocidas y a partir de ellas se realiza el reconocimiento.

Modelo sintáctico Este modelo se basa en encontrar las relaciones estructurales que guardan los objetos de estudio, utilizando la teoría de lenguajes formales. El objetivo es construir una gramática que describa la estructura del universo de objetos.

Redes neuronales Este modelo supone que tiene una estructura de neuronas interconectadas que se estimulan unas a otras, las cuales pueden ser "entrenadas" para dar una cierta respuesta cuando se le presentan determinados valores.

Capítulo 3

Estado del arte

Desde hace tiempo se ha investigado en el área de reconocimiento de patrones ya sea en imagen o audio, y durante los últimos años se han desarrollado tecnologías que la aprovechan, como cámaras con detección de rostros o asistentes personales con procesamiento de lenguaje natural. Cualquier aplicación de ésta en la tecnología se basa en el mismo principio, buscar características de interés y procesarlas hasta que pueda ser clasificada.

3.1 Revisión de trabajos relacionados

En cuanto a señales de audio, siempre se trata de buscar aspectos de la onda que sean comunes como por ejemplo tratar de encontrar repeticiones en la pista. Cuando se habla de reconocimiento de patrones sonoros en pistas musicales significa procesar una o varias señales tratando de visualizar zonas en la pista que sean de interés, como las frecuencias de los bajos o de las percusiones que generalmente describen el tempo.

3.1.1 Detección de tempo

Gran parte del proyecto que se aborda consiste en realizar búsqueda de aspectos rítmicos como el tempo, con los que se pretende hacer comparaciones entre pistas y estimar posiciones de tiempo que sean similares entre éstas.

Un ejemplo es el de Gkiokas et al. [7]; en su trabajo presentan un algoritmo de estimación de tempo y seguimiento rítmico mediante la separación percusivo/armónica de la señal de audio para filtrar características de cada componente de la pista. La estimación rítmica se logra a partir de los picos tomados de la respuesta de un resonador con los cuales se hace una predicción de ritmo con la cual se deduce la pulsación.

Una forma de detectar el tempo es mediante la búsqueda de repeticiones rítmicas donde Smith y Chew [18] presentan sus matrices de autosimilitud, las cuales son útiles para encontrar patrones semejantes y contrastantes en una melodía. Las matrices son creadas por características extraidas de piezas musicales a varias escalas de tiempo para representar la notación musical. Postulan que el peso óptimo de los componentes de la anotación pueden indicar características que puedan ser candidatas en una anotación musical real.

Laroche [13] propone métodos de estimación de tempo como el análisis transitorio el cual consiste en analizar por separado segmentos de frecuencias cuando incrementan de forma brusca, esto con la finalidad de detectar percusiones o bajos, o bien detectar cambios rápidos de espectro en la señal. Todo esto suponiendo que se trata de una pista con un ritmo constante.

Por otra parte Chen et al. [4] presentan el problema del reconocimiento de tempo como un problema de clasificación estadística. Definen cuatro clases de tempo basadas en la percepción humana y no en detección de repeticiónes pues no descubren el

ritmo como tal. Sus clases se definen como: muy lentos, lentos, rápidos y muy rápidos, dependiendo de la cantidad de pulsos por minuto en una pista. Clasifican el tempo de un sonido usando señales monofónicas analizadas en ventanas de tiempo de 30 milisegundos y después combinándolas en grupos de cuatro fragmentos. Un vector final de 128 elementos que consiste en el promedio y varianza de cada grupo es analizado para su posterior clasificación.

D'Aguanno y Vercellesi [6] proponen un sistema de detección de tempo distinto a lo general exclusivo para archivos en formato MP3[1] [15] ya que se basa en la característica de éstos en almacenar cambios en forma de *transformadas de coseno*[2] discretas modificadas. Ésta utiliza un patron de cambio de ventana que hace referencia a cambios en la onda y se encuentra alineada con la tabla de percusiones. No hay análisis de frecuencia, sino que utilizan el patron obtenido de esta línea y un *metrónomo*.

Otro sistema propuesto es el de Hu [10] quien realiza un análisis de autocorrelación envolvente para una señal con la que después crea una función ponderada basada en la autocorrelación para estimar el tempo de una pista. Todo calculado por las amplitudes pico y las posiciones de la autocorrelación en tiempo real. Con ello se estima una aproximación de lo que un humano escucha e interpreta como ritmo en una pista.

Por último se menciona el trabajo de Gulati y Rao [8] quienes utilizan algoritmos de análisis de información de tempo y almacenamiento en bases de datos via aprendizaje automático. Ellos incluyen información conocida de los ritmos existentes para aumentar el desempeño de sus algoritmos. A su finalización comparan los resultados con una base de datos musical. Al descomponer una pieza en varios fragmentos hacen una

[1]MPEG-2 Audio Layer III o MP3 es un formato de audio común usado para música tanto en computadoras como en reproductores de audio.

[2]La *transformada discreta de coseno* es una transformada basada en la transformada de Fourier discreta, pero utilizando œnicamente nœmeros reales.

evaluación utilizando información existente sobre el ritmo comparando y tratando de acertar tiempos en los que hay picos en la onda. Todo funciona como un *mecanismo no supervisado*[3] que busca patrones espectrales en las frecuencias que coincidan con la información proporcionada.

3.1.2 Aplicaciones con el tempo

Una vez que se tiene la información de un tempo aproximado, se puede buscar características en los sonidos que nos permiten generar aplicaciones de todo tipo como generadores de listas de reproducción basadas en el estado de ánimo, generadores de *tonos musicales*[4] o mezcladores automáticos. La meta del proyecto es conseguir generar un sistema que logre mezclar dos sonidos de tal manera que no sea perceptible el cambio entre ellos, o que por lo menos sea un a transición coherente entre pistas.

Una aplicación sería como la que proponen Morman y Rabiner [14] que básicamente es un sistema que reconoce acordes de una pista musical y los clasifica. Analizan pistas musicales en busca de regiones espectrales con diferentes intensidades mediante el uso de filtros para detectar los tonos. Cada tono es clasificado dependiendo de su frecuencia y *escala musical*[5].

La propuesta de Zhang et al. [24] consiste en un sistema capaz de generar un tono de llamada (*ringtone* en inglés) de cualquier canción popular. Su método consiste en localizar las áreas repetitivas de una canción pues éstas son las que generalmente atraen la atención de una persona, después localizan los vocales y comparando con la posición y el tempo estiman un tiempo de inicio y un tiempo de final, todo lo

[3]El *aprendizaje no supervisado* es un mŽtodo de aprendizaje automático donde un modelo es ajustado a las observaciones. Se distingue del aprendizaje supervisado por el hecho de que no hay un conocimiento a priori.
[4]El *tono musical* es la unidad básica de composición musical.
[5]La *escala musical* es la susesión ordenada de todas las notas de un entorno musical.

anterior basándose en reglas *heurísticas*[6]. El sistema puede dar una o varias respuestas dependiendo de la pista.

Los *covers* o versiones[7] se han vuelto algo común en la red. Bertin-Mahieux y Ellis [3] proponen una técnica para detectar si realmente una pista de audio se trata de un cover. Utilizando *cromas* o *llaves de color* comparando el contenido original con el cover, y analizando las diferencias entre ambos creando señales en ciertas posiciones críticas. El resultado muestra posiciones desiguales para una pista que es un cover y posiciones similares para aquellas que usan contenido original.

El trabajo de Wang [21] muestra un framework para extraer tonos y segmentos de música popular, todo completamente sin supervisar. Obteniendo posiciones de un cromagrama usando una mezcla infinita Gaussiana, con las posiciones se extraen series de tiempo de tonos. Con los tonos transforman las posiciones en secuencias de tiempo. Con ambos se construye una cola multidimensional con la que se obtiene un "ritmo armónico".

Rho et al. [16] proponen un sistema de recomendaciones musicales basadas en contexto. Incluye extracción de características, clasificación de humor musical y predicción de emociones humanas. Mediante el uso de *regresión*[8] de *vectores de soporte*[9] se clasifica el humor musical obteniendo un 87.8 % de precisión.

Yi et al. [23] presentan su "Motor de Búsqueda Musical Sensitivo al Tempo", con aplicaciones terapéuticas y de bienestar. Incluye seis modos de interacción: búsqueda por número, búsqueda por aplausos, búsqueda por deslice de tempo, búsqueda por toque de pantalla, búsqueda por audio de muestra y búsqueda por caminata. Para la búsqueda por caminata se obtiene información del acelerómetro para buscar música

[6]Las reglas *heurísticas* se basan en la utilización de reglas empíricas para llegar a una solución.
[7]Un *cover* o versión es la interpretación de una canción grabada por otro artista.
[8]El *análisis de regresión* es un proceso estadístico para la estimación de las relaciones entre variables.
[9]Las *máquinas de soporte vectorial* son técnicas de clasificación y de gran desempeño, comparadas con técnicas clásicas.

con tempo similar al ritmo de caminata y sincroniza la pista con el ritmo que se lleva caminando.

Becker et al. [2] presentan una patente que consiste en un reproductor multimedia interactivo capaz de reconocer el tempo en una canción. Es capaz de reproducir una pieza musical y extraer fragmentos que se pueden combinar en una nueva mezcla. La información musical es obtenida de los discos musicales.

Por último Herberger et al. [9] muestran en su patente un sistema que analiza el sonido y busca por el tempo estimado de una canción reproducida a través de cualquier dispositivo mediante un micrófono. El usuario puede presionar las teclas para generar un tempo constante y alimentar al sistema.

3.2 Análisis comparativo

Los anteriores trabajos presentan características muy similares al proyecto que se plantea en esta tesis. En el cuadro 3.1 se realiza una comparación entre los trabajos seleccionados con mayor relación al proyecto.

Los criterios que se evalúan son los siguientes:

Procesamiento de señales. La inclusión de algún método de procesamiento de señales.

Detección rítmica. Es la detección de alguna característica que defina un ritmo.

Clasificador de patrones. Se evalúa la capacidad de identificar un tipo de patrón.

Procesado en tiempo real. Capacidad de realizar el trabajo de procesamiento mientras se reproduce el contenido analizado.

Datos preprocesados. La necesidad de utilizar datos que contengan información sobre el contenido a analizar.

Listas de reproducción autogeneradas. Es la capacidad de generar listas de reproducción basadas en los datos de procesamiento.

Mezcla inteligente de audio. Es la capacidad de generar mezclas de audio en base a los datos de procesamiento.

El proyecto planteado potencialmente cumple con los criterios de evaluación, ya que se generaron en base a éste. Los diez trabajos que se muestran en el cuadro se seleccionaron pensando también en el proyecto y descartando aquellos cuya implementación es muy similar a alguna otra o su idea no está del todo relacionada con el proyecto.

Los trabajos de Smith y Chew [18] y de Laroche [13] comparten las mismas características; ambos realizan un procesamiento de señales y una detección rítmica. El trabajo de Hu [10] presenta las mismas características con la diferencia de que el procesamiento se hace en tiempo real. Estos trabajos solo muestran los algorítmos necesarios para llegar a una aproximación y no la implementación en algún proyecto.

En los trabajos de Zhang et al. [24] y de Wang [21] se implementa un clasificador de patrones para separar las velocidades de tempo en una pista, de forma similar el trabajo de Chen et al. [4] que además de que implementa un simulador de listas de reproducción que muestra posibilidades de listas basadas en la clasificación previa.

En el trabajo de D'Aguanno y Vercellesi [6] no se procesan señales del todo, la mayor parte de su análisis lo basan el una estructura exclusiva de los archivos MP3 que muestra tener características propias de un ritmo específico.

El trabajo realizado por Bertin-Mahieux y Ellis [3] presenta un procesamiento de señales básico necesario para comparar con datos previamente procesados, donde

buscan diferencias en algunos aspectos para identificar si una pista de audio es una interpretación o es contenido original.

Yi et al. [23] no realizan un procesamiento de señales para su trabajo, solo detectan un ritmo por medio de una entrada de información y buscan pistas de audio que tengan ritmos similares, también genera listas de reproducción de forma autónoma y es capaz de realizar mezclas de sonido, todo en base a datos preprocesados.

Becker et al. [2] no realizan una clasificación de patrones ni generación de listas de reproducción pero logran una mezcla automatizada de audio sin datos preprocesados, lo hacen directamente con una detección de ritmo. En este trabajo sí se realiza un procesamiento de señales en tiempo real.

3.3 Sistema propuesto

Las características evaluadas anteriormente fueron seleccionadas específicamente para que concuerden con las características principales del sistema propuesto.

El sistema es capaz de procesar señales provenientes de los archivos de sonido. Transformando los datos de la pista en datos numéricos comparables con los que se analizan características del sonido. Una de ellas es el ritmo de la pista; en este ámbito el ritmo no describe el tipo de música, sino la información sobre el tempo de la pista.

También se propone la capacidad de clasificar los datos de la pista. Las clasificaciones indican que parte de los datos pertenecen al inicio y final de una pista. Estos son los que sufrirán cambios. Otra de las indicaciones es la disponibilidad entre pistas para crear o no una modificación; entre ciertas pistas no habrá necesidad de crear transiciones entre inicio y final correspondiente si estos son similares en tempo, de lo contrario, se buscaría un tempo entre inicio y final que sea similar para crear una transición.

Cuadro 3.1 – Cuadro comparativo de trabajos relacionados.

Trabajo	**Procesamiento de señales**	**Detección rítmica**	**Clasificador de patrones**	**Procesamiento en tiempo real**	**Datos preprocesados**	**Listas de reproducción autogeneradas**	**Mezcla inteligente de audio**
Smith y Chew [18]	✓	✓	X	X	X	X	X
Laroche [13]	✓	✓	X	X	X	X	X
Chen et al. [4]	✓	✓	✓	X	X	✓	X
D'Aguanno y Vercellesi [6]	≈	✓	✓	✓	≈	X	X
Hu [10]	✓	✓	X	✓	X	X	X
Zhang et al. [24]	✓	✓	✓	X	X	X	X
Bertin-Mahieux y Ellis [3]	✓	X	X	X	✓	X	X
Wang [21]	✓	✓	✓	X	X	X	X
Yi et al. [23]	X	✓	X	✓	✓	✓	≈
Becker et al. [2]	✓	✓	X	✓	X	X	≈
Propuesta de tesis	✓	✓	✓	✓	≈	✓	✓

✓ - Implementado X - No implementado ≈ - Parcialmente implementado

Una pequeña parte del tiempo se realiza un procesamiento en tiempo real; éste sirve para obtener los datos necesarios a ser analizados mientras se reproduce una pista. Estos datos preprocesados se guardan en un archivo de texto que sirve como base de datos para realizar comparaciones entre la pista entrante y la saliente. El análisis consiste en comparar los datos de varios archivos con el actual y encontrar la mejor opción de acuerdo a la pista actual.

La autogeneración de listas de reproducción y la mezcla inteligente de audio son las características principales de acuerdo con el objetivo de la tesis. Realizando comparaciones entre los datos preprocesados se puede generar una lista de reproducción. La mezcla inteligente funciona de manera similar; se comparan los datos entre una pista y otra mientras se reproduce la primera de ellas y entre el inicio y final se produce una transición con la intención de suavizar la pausa entre pistas.

Capítulo 4

Solución propuesta

El objetivo principal de este trabajo es proporcionar una técnica que permita buscar pistas que se relacionen por su ritmo real. Mediante un software programado en Python, se realizan las tareas de analizar archivos de sonido y guardar la información de ritmo en archivos de datos. La implementación de este software consiste principalmente en tres partes; la reproducción, que se lleva acabo en un *hilo*[1] separado que solo se encarga de leer fragmentos de sonido y reproducirlos; el análisis, se realiza en otro hilo que toma información del fragmento actual reproducido y la selección de pista, que corre en el mismo hilo que el análisis y se encarga de seleccionar la mejor pista candidata a ser reproducida.

4.1 Metodología

En el capítulo anterior se mencionan las características principales del proyecto y se hace una comparación con trabajos relacionados. Ahora se discuten los procedimientos necesarios para lograr el funcionamiento propuesto y las herramientas utilizadas para el proyecto.

[1]Un *hilo* es una unidad de procesamiento que permite al software ejecutar tareas en paralelo.

4.1.1 Planeación de proyecto

Para comenzar con el desarrollo del proyecto es necesario hacer una selección de herramientas. Se seleccionó el lenguaje de programación Python[2] debido a su facilidad de uso, con el cual se puede lograr hacer prototipos funcionales en poco tiempo.

Dentro de las librerías utilizadas, se mencionan aquellas que permiten lograr el análisis necesario. Por defecto Python incluye varios módulos que permiten la manipulación de señales provenientes de archivos de sonido.

Python incluye en su librería estandar el módulo wave[3] que hace posible la manipulacion de archivos en formato WAV[4] sin compresión. También incluye el módulo audioop[5] con la cual se puede hacer varias modificaciones a los fragmentos generados por el módulo wave tales como sumar dos fragmentos, aumentar la intensidad de los mismos, entre otras operaciones.

También se utilizaron librerias de terceros que facilitan la tarea de análisis. Para poder reproducir los fragmentos de sonido en un archivo se utilizó la librería PyAudio[6]. Además se utilizó la librería NumPy[7] con la que es posible realizar el cálculo de la transformada rápida de Fourier con la que se obtiene la intensidad de cada frecuencia en una pista.

[2]Python es un lenguaje de programación propósito general de alto nivel.
[3]El módulo wave proporciona una interfaz conveniente para la manipulación de archivos WAV.
[4]WAV es un formato de audio digital generalmente sin compresión.
[5]El módulo audioop proporciona operaciones para la manipulación de fragmentos de audio.
[6]PyAudio es una librería de Python para la reproducción y grabación de flujos de audio
[7]NumPy es una librería de Python que agrega mayor soporte para matrices y vectores.

4.1.2 Método de procesamiento

La intención del procesameinto de señales en este proyecto es obtener el tempo de la pista en reproducción. Para ello primero se procede a leer un archivo de sonido, en este caso se optó por el formato WAV sin compresión debido a su fácil manipulación sin la necesidad de utilizar librerías de decodificación, sin embargo el proceso es facilmente adaptable a cualquier formato.

El modulo wave permite la lectura de archivos en formato WAV de los cuales se obtiene una serie de fragmentos que corresponden a un instante de sonido. Mediante la librería PyAudio se crea un flujo de lectura de estos fragmentos, esto reproduce el instante de sonido. El fragmento reproducido esta en formato PCM[8] que describe la forma de la onda producida, con el fragmento es posible analizar una aproximación a la frecuencia independiente de cada sonido mediante el cálculo de una transformada rápida de Fourier.

Con las frecuencias obtenidas se procede a calcular una aproximación de tempo para lograr una detección de ritmo. En este contexto el tempo calculado no es fiel al término real; para este contexto el tempo hace referencia a los sonidos reproducidos en el instante y no al instrumento específico como se hace normalmente. La aproximación se logra calculando una diferencia entre las frecuencias actuales y las frecuencias del instante pasado; si la diferencia en una banda es muy amplia, ésta influenciará la decisión de tomar el instante como un pulso o no. Finalmente la diferencia de tiempo entre pulsos define el tempo aproximado para el instante.

[8]La modulación por impulsos codificados (PCM del inglés Pulse Code Modulation) es un procedimiento para transformar una señal analógica a secuencias de bits.

4.1.3 Método de clasificación

Los datos preprocesados de los archivos de sonido describen un ritmo algo específico para el tipo de música que se reproduce. Éstos contienen un inicio y un final casi siempre distintos, sin embargo es necesario detectar en que parte de la pista deja de ser el inicio y en que parte comienza el final de la misma. Por ello es necesario hacer una clasificación de datos.

Con una clasificación de datos es posible comparar inicio y final de dos pistas distintas pero que comparten similitudes en sus ritmos, estas similitudes son encontradas mediante un algorítmo de reconocimiento de patrones, en el que se analizan los datos preprocesados para buscar zonas en la pista que tengan un tempo que siga un patrón parecido en ambas pistas. De esta manera es posible difuminar el final de una pista donde el tempo es tentativamente distinto con el inicio de otra pista donde el tempo se iguala.

Los datos preprocesados consisten en arreglos de valores que representan tiempo en la pista e intensidad de pulso para el tiempo dado. El programa analiza los datos y compara cada valor de intensidad buscando repeticiones en su comportamiento; es decir, cuando los valores de intensidad suben y bajan en cierta medida respetando un rango especificado.

4.2 Diseño e implementación

En un principio se tenía una combinación de elementos de interfaz con elementos algorítmicos; ciertos eventos de la interfaz controlaban algunas variables que el algoritmo utilizaba. Ahora el software trabaja con módulos independientes, lo que inclusive permite implementar algunos de ellos en un lenguaje de programación

distinto. Principalmente el software se compone por módulos de interfaz que controlan los eventos de la misma y módulos del algoritmo que simplemente ejecutan una tarea hasta terminarla, ademas se implementan conectores entre los módulos de interfaz y los del algoritmo con el fin de comunicar a ambos de forma que no existan interferencias ni pérdidas de datos.

4.2.1 Interfaz

En la figura 4.1 se muestra el diseño propuesto de la interfaz de acuerdo con el objetivo principal de la tesis.

Figura 4.1 – La interfaz de usuario utiliza controles simples para evitar complicaciones de uso.

El software tiene una interfaz simple, consta de un botón de reproducción/pausa y un par de botones para la pista siguiente y la anterior. Muestra una etiqueta para el nombre del archivo en reproducción y una para el tiempo restante. También se tiene un par de botones para activar la generación de listas de reproducción automática y la reproducción sin pausa.

4.2.2 Funcionamiento interno

Las funciones principales del software son tres:

Reproducción de archivos. La función básica es la de reproducir archivos en formato WAV sin compresión.

Generación de listas de reproducción. Un generador de listas basado en la inforación obtenida del procesamiento de cada archivo. Mediante un algoritmo de búsqueda de similitudes entre los datos se puede encontrar la siguiente pista a reproducir.

Transición inteligente. Al estar activa la opción de reproducción sin pausa, las pistas tendrán una trnasición suave entre inicio y fin. La selección de los puntos de transición se hace mediante un algoritmo que reconoce un punto entre dos pistas que tiene un ritmo que sea semejante.

En la figura 4.2 se muestra un diagrama del funcionamiento del software.

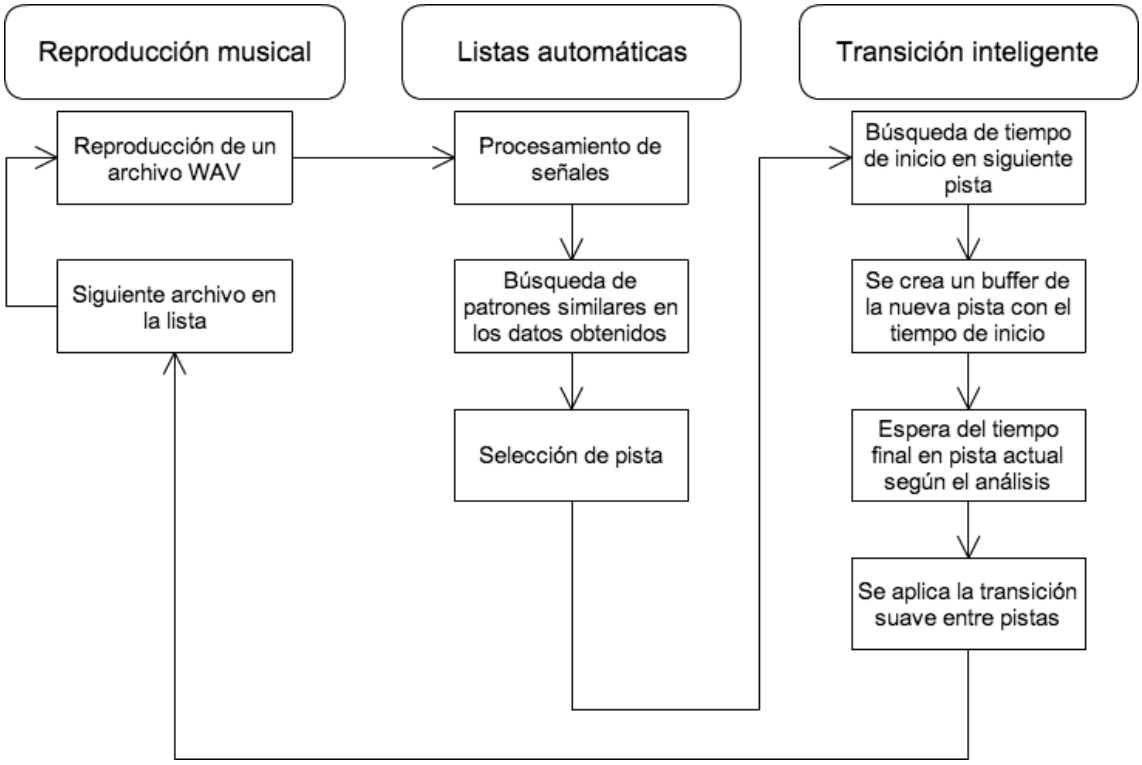


Figura 4.2 – Funciones principales del software.

4.3 Obtención de información relevante

Para poder generar una lista de reproducción basada en el ritmo de las pistas es necesario procesar éstas antes. Para ello es necesario utilizar la transformada rápida de Fourier con la cual se obtiene información de las frecuencias individuales en un tiempo t. El proceso de obtención de información relevante consiste en capturar en ventanas de tiempo constantes la información de las frecuencias, al término de cada ventana de tiempo se hace una selección lógica por medio de votaciones. Los valores en tiempo t_1 mayores a los del tiempo t_0 dan un voto positivo, los menores dan un voto negativo, al final se cuentan los votos de la ventana de tiempo y si la mayoría son positivos entonces se almacena el valor actual, de no serlo simplemente se continua a la próxida ventana de tiempo.

La lógica anterior se basa en que la pista presenta intensidades propias de su ritmo, la intención es capturar los momentos en que ocurre un cambio de intensidades para comparar con otras pistas y encontrar aquellas que sean compatibles en ritmo.

Ya que al reproducir una pista es necesario obtener de ella sus datos en formato PCM, éstos se utilizan para realizar el proceso de obtención de información en forma paralela a la reproducción, cada instante de tiempo se actualizan estos datos, de esa manera se asegura que el tiempo de obtención sea equivalente al de la pista. Los datos obtenidos se almacenan en un archivo de texto plano.

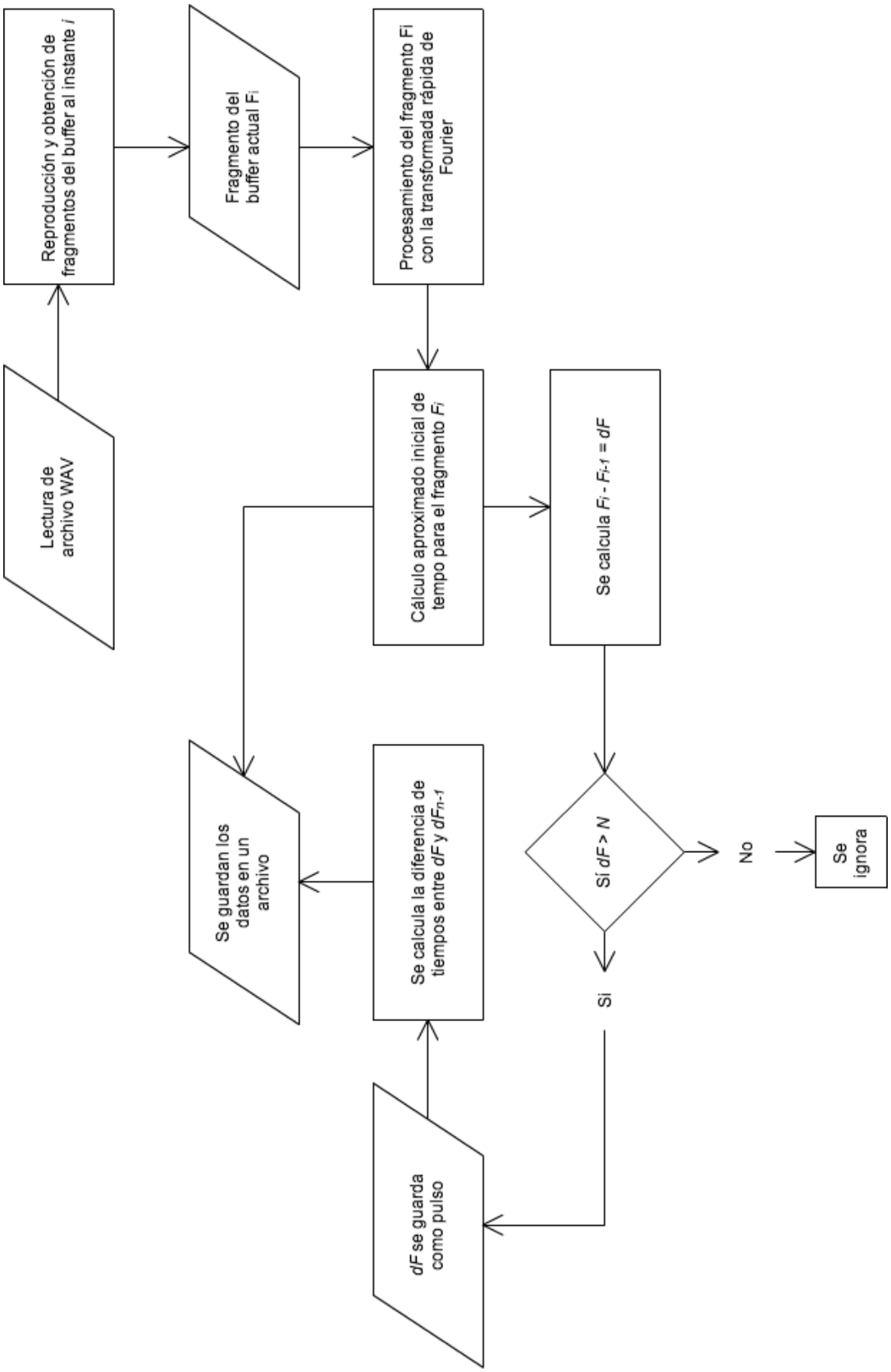


Figura 4.3 – Obtención de información relevante de la pista.

4.4 Generación de listas de reproducción

La selección de pistas se realiza mediante un análisis de datos que corre en paralelo con la reproducción actual. Cada pista que se reproduce tiene un archivo de información obtenida y se utiliza para comparar la pista correspondiente con las pistas en el directorio. La rutina que compara los datos entre pistas consiste en buscar un patrón de comportamiento en los datos, primero se buscan patrones en el archivo de datos de la pista en reproducción, es decir, buscar secciones en los datos que sean coincidentes, se almacena el patrón y se guardan los tiempos en que ocurre el patrón encontrado. Con esa información se procede a analizar los archivos de datos de las pistas en el directorio, por cada análisis se realiza una rutina de comparación entre los datos de la pista actual p_r y los de la candidata p_c. El análisis consiste en comparar los datos de ambas pistas de tal manera que sigan un patrón de subidas y bajadas en los valores de los datos; si el valor de p_{ri} y el valor de p_{ci} son al mismo tiempo mayores o menores que p_{ri-1} y p_{ci-1} respectivamente entonces los patrones son compatibles y se procede a comparar los siguientes valores, en caso de que p_{ri-1} sea mayor que su valor anterior y p_{ci-1} menor al suyo entonces el patrón ya no es aceptable si se procede a analizar a otro patrón.

Con esa información se obtienen los tiempos donde hay patrones compatibles entre la pista actual y la candidata, ahora solo queda seleccionar el que mejor se adapte. Normalmente cada pista tiene varios patrones candidatos, el mejor de ellos debe tener un equilibrio entre posición temporal en ambas pistas y de precisión entre patrones de la pista actual y la candidata. La pista seleccionada debe de cumplir los mismos criterios, pero esta vez entre los patrones seleccionados de cada pista anteriormente.

La primera vez que se utiliza una pista no es posible obtener su información hasta que se haya terminado de reproducir, por lo que se utiliza la información obtenida

hasta el momento para seleccionar una pista compatible en ritmo sin pasar por todos los filtros.

Si por alguna razón no se encuentra una pista ya sea por que no es compatible o por algún error interno, entonces se selecciona una pista al azar.

La lista de reproducción se va generando conforme avanza la reproducción, para evitar repeticiones, se tiene una *lista tabú*[9] que se actualiza con las pistas previas hasta que se tenga una lista con todas las pistas del directorio actual.

[9]Una *lista tabú* es una lista que se guarda en memoria durante un proceso para almacenar datos del mísmo con la intención de evitar repetirlos.

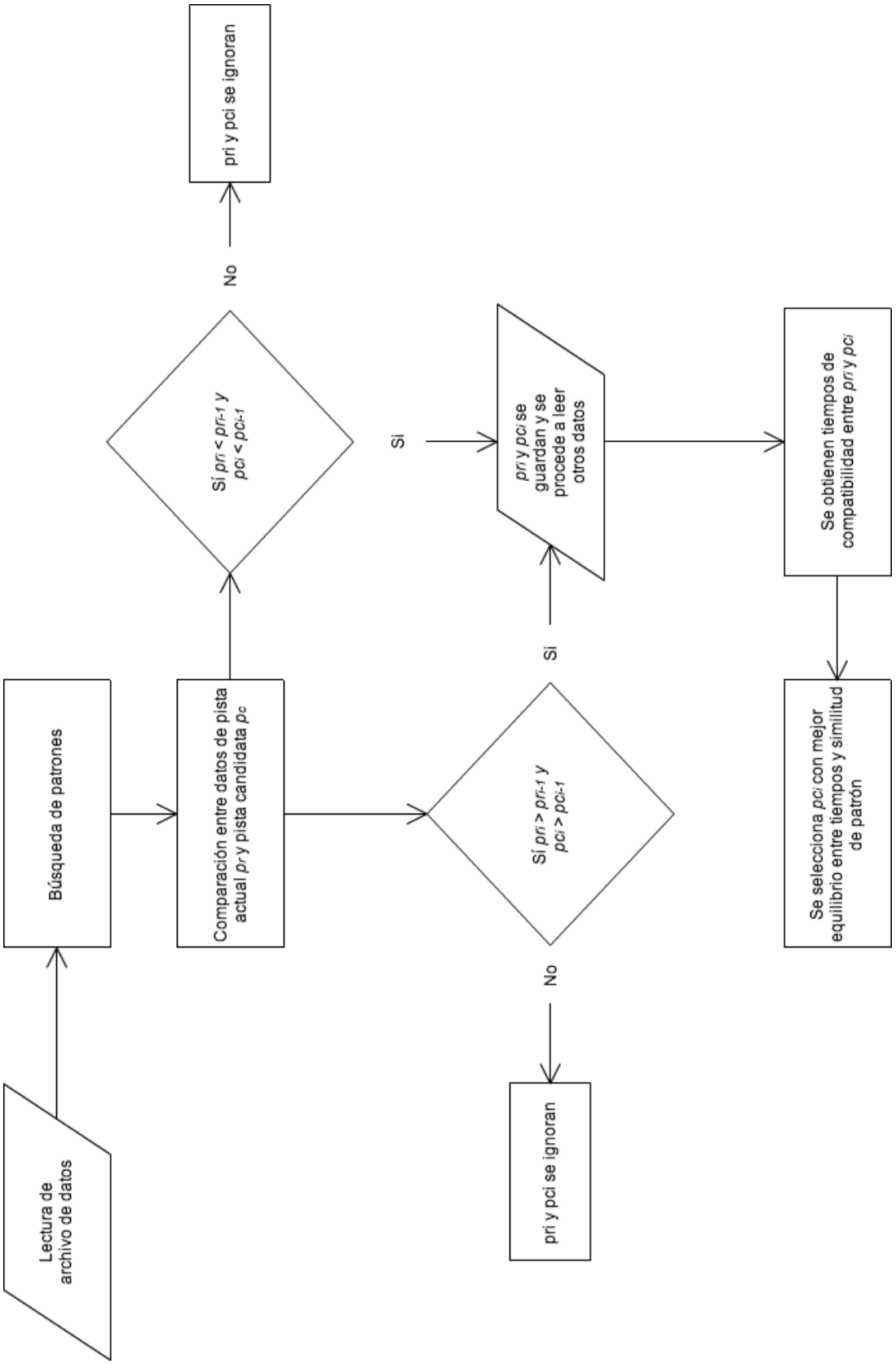


Figura 4.4 – Diagrama que muestra la forma en que se generan las listas.

4.5 Transición inteligente entre pistas

Una de los datos obtenidos del análisis de la pista seleccionada es el tiempo en que ocurre el patrón encontrado, tanto en la pista actual como en la seleccionada, este dato describe el instante en la pista actual en que se puede iniciar una transición de salida para la pista actual combinado con una de entrada para la nueva pista, de la cual también se conoce el tiempo en que ocurriría la transición.

4.6 Dificultades

Durante el desarrollo del proyecto presentaron varias dificultades; la búsqueda un método eficiente para obtener datos de procesamiento, el intercambio de información entre hilos sin afectar el flujo principal, efectuar transiciones sin perder información de procesamiento entre pistas, entre otros problemas menores.

El principal problema fue el manejo de hilos en el programa; al realizar distintas actividades en el sistema operativo se generaban pequeñas pérdidas de datos y pausas en la reproducción que afectaban al análisis. La solución fue utilizar subprocesos que corren de forma independiente del programa en segundo plano para evitar interferencias. Los subprocesos son programas independientes por lo tanto se pueden escribir en cualquier lenguaje distinto a la implementación principal, lo que aumenta la eficiencia en general.

Capítulo 5

Evaluación

Para un completo desarrollo de software es necesario realizar pruebas de software, con los cuales es posible detectar errores, defectos o vulnerabilidades que se pueden reparar para ofrecer un mejor funcionamiento. En este caso se realizan dos tipos de pruebas; las pruebas de rendimiento, que consisten en pruebas que se ejecutan por si solas y que arrojan un resultado cuantificable y pruebas con usuarios, en las cuales se les pide a los usuarios realizar ciertas acciones con el programa y evaluar tales acciones.

5.1 Diseño experimental

Para el funcionamiento adecuado del software se utilizaron varios métodos que logran un flujo ininterrumpido pero que requieren más recursos. Para evaluar el rendimiento del software bajo éstas circunstancias, se toman en cuenta los siguientes criterios:

Velocidad de procesamiento Se evalúa la velocidad que el software necesita para realizar las operaciones necesarias. Dependiendo de la plataforma y el hardware en que se ejecuta el software podrían existir diferencias de velocidad.

Precisión algorítmica El software genera datos en base a información obtenida del procesamiento. La modificación de uno o varios parámetros en los algoritmos genera datos distintos. En este caso se evalúan las condiciones en las que opera el software y que tan precisos son los datos generados.

Eficiencia de ejecución El procesamiento, análisis y reproducción trabajan en tiempo real, representan una carga de memoria. Para este criterio se evalúa la eficiencia en que trabaja el software en distintas condiciones de plataforma y hardware, así como el uso de hilos.

Estas pruebas se realizan mediante rutinas que ejecutan métodos específicos del software o ejecución completa, siempre con parámetros cambiantes. Se llevan a cabo en plataformas distintas bajo circunstancias de hardware variadas (ejecución única, ejecución junto a otros programas, etcétera).

Adicionalmente a las pruebas de rendimiento se realizan pruebas básicas con usuarios donde se evalúan los siguientes criterios:

Generación automática de listas de reproducción De forma similar a la precisión algorítmica, pero desde el punto de vista de los usuarios. La intención es evaluar el grado en que se seleccionan las pistas y la fidelidad del ritmo musical.

Transición y mezclado entre pistas En este criterio se evalúa el grado de aceptación por parte de los usuarios de la función de mezclado inteligente utilizando efectos de transición entre pistas.

5.1.1 Velocidad de procesamiento

Esta prueba consiste en experimentar con el tiempo que le toma al software procesar los archivos para una sola pista. Se realizara de dos modos: sin reproducción

y con reproducción. Cuando se realiza sin reproducción, el software no tomará en cuenta la pista sino que pasará directo al procesamiento de datos y se ejecutará hasta acabar por completo con todos los archivos. Cuando se realiza con reproducción, el software utiliza un limite de archivos para asegurar terminar antes de que la pista actual finalice.

Para cada uno de los modos se hacen de 20 a 30 repeticiones y en el caso de el modo sin reproducción se utilizan de 50 a 100 archivos de prueba.

5.1.2 Precisión algorítmica

Muchos de los elementos dentro del algoritmo son constantes que se han elegido en base al comportamiento buscado, sin embargo, muchas de estas constantes parecen ser más favorables en ciertas ocasciones. En esta prueba se experimenta con la precisión de los algoritmos cuando sus constantes son cambiadas, para medir la precisión se utiliza una lista de reproducción modelo, la cual consiste en pistas que van cambiando sus bpm de forma progresiva. Se realizan de 20 a 30 repeticiones por cada cambio.

5.1.3 Eficiencia de ejecución

Con las pruebas anteriores ahora se tiene la mejor configuración de funcionamiento para el software, ahora se experimenta con ésta la eficiencia de ejecución donde se evalúa la cantidad de recursos mínima y máxima que necesita el software para funcionar. De igual manera se realizan de 20 a 30 repeticiones con y sin reproducción y para el modo sin reproducción se utilizan de 50 a 100 archivos de prueba.

5.1.4 Generación automática de listas de reproducción

Las pruebas con usuarios son más enfocadas al uso del software y no al funcionamiento interno. En el caso de la prueba de generación de listas de reproducción el usuario evalúa desde su punto de vista que eficiente es el software en cuanto a selección de pistas. Se utilizan pistas seleccionadas por el propio usuario y al final se le hará una encuesta preguntándole sobre la eficiencia de la generación automática de listas de reproducción.

5.1.5 Transición y mezclado entre pistas

Durante la prueba se le pide al usuario que active la función de transición y mezclado y que escuche fragmentos de pistas. El usuario evalúa esta función dando a conocer que tan eficiente es y si es de su agrado.

5.2 Resultados

Después de realizar las pruebas, se evalúan los resultados obtenidos. En el caso de las pruebas de rendimiento, se generaron *scripts*[1] que corren solo ciertas partes del programa y guardan información. Las pruebas con usuarios fueron independeintes, a cada usuario se le dieron tareas a realizar con el programa y evaluarlas al término de cada una.

[1]Un *script* es un conjunto de instrucciones interpretadas por el sistema operativo para realizar tareas simples.

5.2.1 Pruebas de rendimiento

En total se realizaron tres pruebas de rendimiento. Cada prueba se realiza en las plataformas Linux[2], Windows[3] y OS X[4] (excepto la prueba de presición algorítmica), en cada plataforma se realiza la prueba sin procesos en ejecución (correr el programa con la mayor cantidad de memoria disponible) y con varios procesos en ejecución (correr el programa solo con memoria suficiente). La cantidad de repeticiones depende de cada prueba, rondando entre las 50 y 100 repeticiones.

La prueba de velocidad de procesamiento se efectúa tomando los tiempos que tarda el programa para generar los datos, procesarlos, compararlos y dar una respuesta; en este caso la respuesta es la siguiente pista a reproducir.

En el caso de la prueba de precisión algorítmica se modifican aspectos internos; como un umbral de reconocimiento utilizado para suavizar los valores y hacer más facil el reconocimiento, o el número de bandas de frecuencia que afectan la precisión con la que se procesa la información. El porcentaje de aserción se evalúa con una lista de reproducción predeterminada; entre más fiel sea la lista de reproducción generada a la predeterminada, mayor será su porcentaje de aserción.

Utilizando la mejor configuración de la prueba de precisión algorítmica, se efectúa una prueba de eficiencia de ejecución, simplemente se ejecuta el programa bajo distintas circunstancias y se toman los datos directamente del sistema operativo.

[2]Linux es un sistema operativo de libre distribución basado en Unix (otro sistema operativo de libre distribución), desarrollado originalmente por Linus Torvalds.
[3]Microsoft Windows es el nombre de una familia de sistemas operativos desarrollados y vendidos por Microsoft.
[4]OS X, antes llamado Mac OS X, es una serie de sistemas operativos basados en Unix desarrollados, comercializados y vendidos por Apple Inc. que ha sido incluido en su gama de computadoras Macintosh desde el año de 2002.

5.2.2 Pruebas con usuarios

Para las pruebas con usuarios se realizará una sola ejecución por usuario y al final se le pedirá llenar una encuesta. La cantidad mínima de usuarios será 20 (10 mujeres y 10 hombres, 10 jóvenes y 10 adultos).

Para la prueba de generación automática de listas de reproducción, se dió a los usuarios la tarea de escuchar una de las pistas disponibles y evaluar el nivel en que el programa selecciona automáticamente las pistas.

Después de la evaluación anterior, los usuarios evaluaron los efectos de transición y mezclado entre las pistas.

Los datos reales aún no se tienen, se pretende tenerlos listos entre finales de Junio y mitad de Julio, tiempo en el que se pretende tener las pruebas completas y tener las suficientes encuestas. A continuación se presentan los cuadros y figuras donde estarán los resultados reales pero con valores provisionales que no reflejan nada.

Cuadro 5.1 – Comparativa de velocidad de procesamiento usando distintos parametros tanto de hardware como de software. (Datos provisionales)

Sistema operativo	**Tiempo de procesamiento sin reproducción**	**Tiempo de procesamiento con reproducción**
Junto a varios procesos		
Mac OS X	0.0 s	0.0 s
Ubuntu	0.0 s	0.0 s
Windows	0.0 s	0.0 s
Proceso único		
Mac OS X	0.0 s	0.0 s
Ubuntu	0.0 s	0.0 s
Windows	0.0 s	0.0 s

Cuadro 5.2 – Se compara la precisión con la que el programa acierta en la selección automática de pistas. (Datos provisionales)

Umbral de reconocimiento	**Número de bandas de frecuencia**	**Porcentaje de aserción sin reproducción**	**Porcentaje de aserción con reproducción**
Sin umbral	4	0.0	0.0
	8	0.0	0.0
	16	0.0	0.0
5 unidades	4	0.0	0.0
	8	0.0	0.0
	16	0.0	0.0
10 unidades	4	0.0	0.0
	8	0.0	0.0
	16	0.0	0.0

Cuadro 5.3 – Se comparan los resultados obtenidos y se analizan los recursos utilizados en cada sistema operativo. (Datos provisionales)

Sistema operativo	**Recursos utilizados por el sistema operativo sin reproducción**	**Recursos utilizados por el sistema operativo con reproducción**
Junto a varios procesos		
Mac OS X	0.0	0.0
Ubuntu	0.0	0.0
Windows	0.0	0.0
Proceso único		
Mac OS X	0.0	0.0
Ubuntu	0.0	0.0
Windows	0.0	0.0

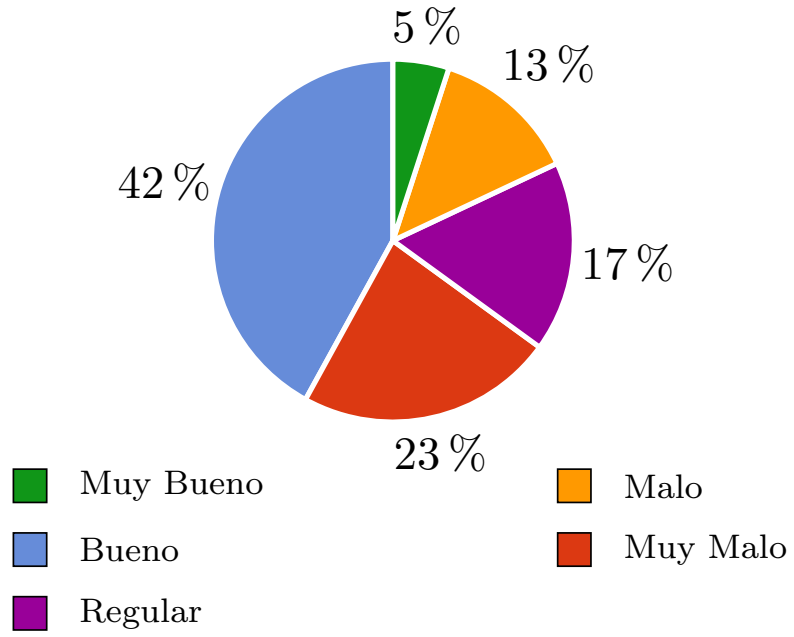


Figura 5.1 – Nivel de fidelidad de generación automática de listas según los usuarios. (Datos provisionales)

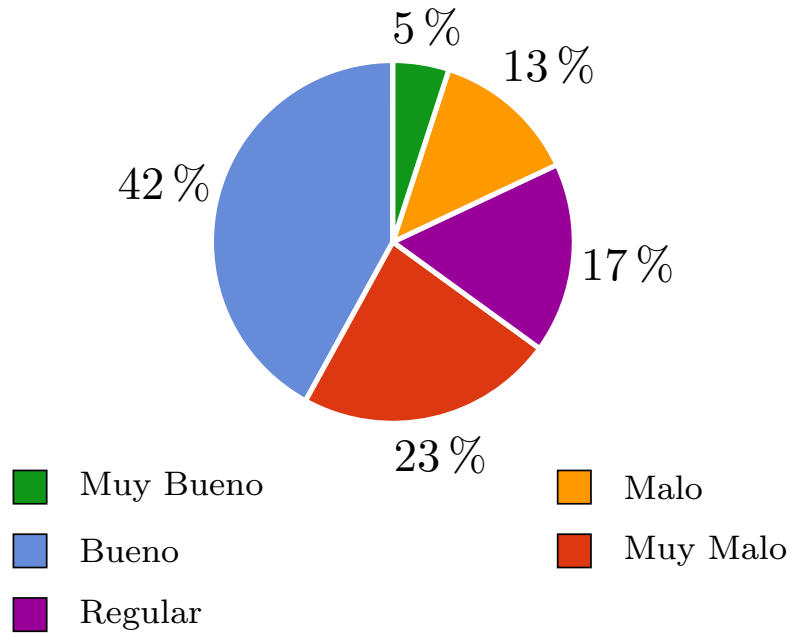


Figura 5.2 – Nivel de ejecución de efectos de transición y mezclado según los usuarios. (Datos provisionales)

5.3 Análisis de resultados

Con el análisis de resultados se obtienen respuestas que describen el funcionamiento del software en situaciones distintas. La modificación de ciertas variables genera un flujo de ejecución muy distinto, algunas veces favorable para la misma.

A continuación se muestra el análisis de cada prueba, los factores cambiantes en cada una y el objetivo de su realización.

5.3.1 Análisis de rendimiento

El cuadro 5.1 muestra los resultados de la prueba de velocidad de procesamiento, ésta fue realizada para conocer el tiempo que necesita el software para dar una respuesta. Las pistas de audio tienen longitudes variables y gran parte de las operaciones del software dependen de ésta; el análisis principal consiste en dividir la pista en bloques de tamaño constante y el número de bloques depende de la longitud de la pista, la detección de patrones consiste en analizar archivos generados por la operación anterior y su tamaño es proporcional a la longitud de la pista. Si una pista es de gran longitud entonces tardará más en realizar el procesamiento, por ello es necesario realizar todas las tareas involucradas en un hilo distinto al principal. La intención de esta prueba es asegurar que el hilo de procesamiento termina antes que la reproducción para tener una respuesta a la siguiente pista.

Las pruebas iniciales muestran que la velocidad de procesamiento es afectada por la reproducción de la pista, es por eso que se evalúa la velocidad que toma el procesamiento con y sin reproducción de pista. ésto es debido a que la reproducción también ocurre en un hilo distinto al principal y toma parte de los recursos para ejecutarse.

En el cuadro 5.2 se muestran los resultados de la prueba de precisión algorítmica que consiste en analizar las respuestas obtenidas del software bajo ciertas condiciones cambiantes con la intención de que los datos permanezcan sin cambios. En caso de que existan cambios se toma como mejor condición aquella que tenga cambios mínimos.

Para esta prueba se tiene una lista de reproducción estática como modelo que contiene pistas en una sucesión que sigue una lógica, tal sucesión consiste en pistas de prueba que incrementan sus bpm conforme avanza en la lista. Uno de los factores para esta prueba es el umbral de reconocimiento, éste es un valor constante que suaviza los valores obtenidos del procesamiento para tener un mejor rango de reconocimiento de bpm, modificar este valor puede conllevar a cambios en la forma en que se genera el archivo de análisis y por lo tanto en la forma de detectar patrones en las pistas. Otro factor a tomar en cuenta es la cantidad de bandas utilizadas, entre mayor sea la cantidad menos sensible es el software pero más rápida es su ejecución, para la prueba de velocidad se utilizaron 8 bandas, para esta prueba se utilizan 4, 8 y 16 bandas. Por último se ejecuta la prueba con y sin reproducción para analizar si ésto afecta la forma en que se genera una respuesta.

En el cuadro 5.3 se muestra la evaluación de la prueba de eficiencia de ejecución, ésta sirve para conocer la cantidad de recursos que utiliza el software al ejecutarse en distintos entornos y bajo condiciones de mucha o poca memoria. La intención es llevar al límite al software y saber hasta que punto aún puede continuar operando de forma constante como en las pruebas anteriores. Es importante saber cual es el requerimiento mínimo y máximo del software para asegurar el mejor funcionamiento posible y no interferir el la ejecución de otros procesos.

5.3.2 Análisis con usuarios

En los resultados de la prueba de generación automática de listas de reproducción mostrados en la figura 5.1, los usuarios tienen la tarea de evaluar el seguimiento rítmico de la lista de reproducción.

Para la prueba se utiliza una lista pregenerada con pistas de prueba y una lista de reproducción que se generará en el momento. Al final de la prueba se le pedirá al usuario calificar la fidelidad musical; es decir, si el software logró generar una lista de acuerdo a los ritmos.

Esta prueba es muy variable en sus resultados y depende de los gustos de cada usuario, pero es importante conocer estos datos para saber si el software realmente cumple con su función.

En la figura 5.2 se muestran los resultados de la prueba de transición y mezclado entre pistas, ésta pone a los usuarios en la tarea de evaluar la función del software para generar una transición entre dos pistas. Para ello se reproduce una lista pregenerada y una generada en el momento. Los usuarios evalúan si la función se ejecuta de forma adecuada.

Capítulo 6

Conclusiones

Este capítulo necesita datos de la evaluación, por lo tanto está sujeto a cambios dependiendo de los resultados. Se marca en rojo lo que tenga más probabilidad de ser cambiado.

Durante el desarrollo de este proyecto se presentó un software con la capacidad de analizar pistas de audio para obtener información del ritmo y de esa manera generar listas de reproducción basadas en tal característica. La finalidad de la herramienta es brindar una mejor experiencia de reproducción musical para el usuario.

Se implementaron métodos de análisis de señales y detección de patrones, los cuales permiten detectar la velocidad del ritmo e inicio y final de pista con los que se generan listas de reproducción de forma automática.

Para producir una mejor experiencia de reproducción musical se implementaron efectos de transición entre pistas; entre dos pistas se calcula un momento temporal en el que los ritmos son similares, se aplica un efecto de difuminación en la primera pista que disminuye su potencia, mientras la segunda pista la aumenta con un efecto similar.

El desarrollo principal se realizó en el lenguaje de programación Python mientras que algunas rutinas de procesamiento se realizaron en el lenguaje de programación Java debido a la carga algorítmica.

6.1 Discusión

El prototipo de software desarrollado demuestra que sus resultados son favorables en cuanto a su objetivo, de acuerdo con los usuarios que lo probaron. La generación de listas respeta el ritmo de las pistas y se crea una lista con una sucesión lógica de pistas.

Aunque las transiciones entre pistas son efectos que intentan mejorar la experiencia del usuario, no a todos les convence esta característica, algunos usuarios prefirieron desactivar la característica, a otros usuarios les agradó la idea pero piensan que se puede mejorar la detección del tiempo correcto en que se debe realizar.

De acuerdo con los usuarios, la interfaz no está muy elaborada, pero es sencilla de utilizar pues sus controles se reconocen a simple vista y sus opciones se pueden identificar unas de otras facilmente.

Aunque los resultados de ejecución del software son los esperados, existen problemas de velocidad de procesamiento que causan conflictos entre los hilos del software; el procesamiento en tiempo real se vuelve muy pesado en función al número de archivos de sonido en un directorio. Debido a que la restricción de tiempo se define por la duración de la pista actual, un alternativa viable a la ejecución en tiempo real de este procesamiento es ejecutar esta rutina como un servicio transparente al usuario que constantemente analiza los archivos y genera listas de reproducción de acuerdo a los parametros definidos por el usuario.

El software necesita una grán cantidad de recursos para ejecutarse, esto es debido a que el manejo de hilos no es lo más eficiente posible, además existe una probabilidad de que las rutinas se pausen temporalmente, lo que genera una pequeña pérdida de datos que podría generar cambios en los resultados. Lo mejor en este caso sería

reescribir las rutinas utilizando métodos más eficientes.

6.2 Trabajo a futuro

Definitivamente hay mucho trabajo por hacer para lograr tener un producto completamente funcional y que esté en condiciones de hacerlo público. Aunque los experimentos no muestran los resultados más favorables, se trata de un prototipo que demuestra un buen funcionamiento que puede mejorar en varios aspectos.

Los cambios inmediatos en el software serían cosas internas, como mejorar los algoritmos y rutinas en general. Agregar nuevas funciones de acuerdo a las tendencias actuales que hagan más atractiva la idea de este software y agregar opciones de personalización. Tambien trabajar en una interfaz gráfica más elaborada que sea amigable para el usuario.

Lo mejor será cambiar el lenguaje de programación, ya que Python no ofrece el desempe–o buscado para este tipo de programas. Probablemente la implementación completa se haga en C/C++ o Java.

Para el futuro se podría trabajar en una versión móvil o probablemente web; ya que las tendencias indican que en el futuro las personas preferirán almacenar sus archivos en la nube o usar servicios de reproducción musical en linea.

Bibliografía

[1] Hussein Baher. *Analog and Digital Signal Processing*. John Wiley & Sons, Inc., Nueva York, NY, EE. UU., 1990. ISBN 978-0-471-92342-8.

[2] Friedemann Becker, Thomas Holl, Michael Kurz, Toine Diepstraten y Daniel Haver. Automatic recognition and matching of tempo and phase of pieces of music, and an interactive music player. 2010. US Patent App. 12/565,766.

[3] Thierry Bertin-Mahieux y Daniel Ellis. Large-scale cover song recognition using hashed chroma landmarks. En *Proceedings of the 2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, págs 117–120. IEEE, Washington, DC, EE. UU., 2011. doi:10.1109/ASPAA.2011.6082307.

[4] Ching-Wei Chen, Kyogu Lee y Ho-Hsiang Wu. Towards a class-based representation of perceptual tempo for music retrieval. En *Proceedings of the 2009 International Conference on Machine Learning and Applications*, págs 602–607. IEEE, Washington, DC, EE. UU., 2009. doi:10.1109/ICMLA.2009.54.

[5] Armando Astarloa Cuéllar y Aitzol Zuloaga Izaguirre. *Sistemas de procesamiento digital*. Delta Publicaciones, Madrid, España, 2008. ISBN 978-84-92453-03-0.

[6] Antonello D'Aguanno y Giancarlo Vercellesi. Tempo induction algorithm in MP3 compressed domain. En *Proceedings of the International Workshop on*

Workshop on Multimedia Information Retrieval, MIR '07, págs 153–158. ACM, Nueva York, NY, EE. UU., 2007. doi:10.1145/1290082.1290105.

[7] Alexandros Gkiokas, Vassilis Katsouros, Elias G. Carayannis y Themos Stafylakis. Music tempo estimation and beat tracking by applying source separation and metrical relations. En *Proceedings of the 2012 IEEE Workshop on Acoustics, Speech and Signal Processing*, págs 421–424. IEEE, Washington, DC, EE. UU., 2012. doi:10.1109/ICASSP.2012.6287906.

[8] Sankalp Gulati y Preeti Rao. Rhythm pattern representations for tempo detection in music. En *Proceedings of the 1st International Conference on Intelligent Interactive Technologies and Multimedia*, IITM '10, págs 241–244. ACM, Nueva York, NY, EE. UU., 2010. doi:10.1145/1963564.1963606.

[9] Tilman Herberger, Titus Tost y Georg Flemming. System and method of bpm determination. 2011. EP Patent 1,377,959.

[10] Jianling Hu. Real-time perceptual tempo estimation for music signal based on envelope autocorrelation. En *Proceedings of the Wireless 2010 International Conference on Communications and Signal Processing*, págs 1–4. IEEE, Washington, DC, EE. UU., 2010. doi:10.1109/WCSP.2010.5633730.

[11] Anil K. Jain, Robert P. W. Duin y Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[12] Josef Kittler. Modelo de sistema de reconocimiento de patrones. En *Notas del seminario de Reconocimiento de Patrones de Grupo de Tratamiento de Imágenes del Instituto de Ingeniería Eléctrica, basado en las notas del curso del Prof. J. Kittler en la Univ. de Surrey*, págs 1–5. 2002.

[13] Jean Laroche. Estimating tempo, swing and beat locations in audio recordings. En *Proceedings of the 2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, págs 135–138. IEEE, Washington, DC, EE. UU., 2001. doi:10.1109/ASPAA.2001.969561.

[14] Joshua Morman y Lawrence Rabiner. A system for the automatic segmentation and classification of chord sequences. En *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, AMCMM '06, págs 1–10. ACM, Nueva York, NY, EE. UU., 2006. doi:10.1145/1178723.1178725.

[15] Martin Nilsson. The audio/mpeg media type. RFC 3003, IETF Secretariat, 2000.

[16] Seungmin Rho, Byeong-jun Han y Eenjun Hwang. Svr-based music mood classification and context-based music recommendation. En *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, págs 713–716. ACM, Nueva York, NY, EE. UU., 2009. doi:10.1145/1631272.1631395.

[17] Davide Rocchesso. *Introduction to Sound Processing*. Associazione Culturale Mondo Estremo, Verona, Italia, 2004. ISBN 978-88-901126-1-4.

[18] Jordan B.L. Smith y Elaine Chew. Using quadratic programming to estimate feature relevance in structural analyses of music. En *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, págs 113–122. ACM, Nueva York, NY, EE. UU., 2013. doi:10.1145/2502081.2502124.

[19] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, EE. UU., 1997. ISBN 978-0-9660176-3-2.

[20] James S. Walker. *Fast Fourier Transforms, Second Edition*. Studies in Advanced

Mathematics. Taylor & Francis, Abingdon, Oxford, Reino Unido, 1996. ISBN 978-0-849-37163-9.

[21] Yun-Sheng Wang. Toward segmentation of popular music. En *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, ICMR '13, págs 345–348. ACM, Nueva York, NY, EE. UU., 2013. doi:10.1145/2461466.2461534.

[22] Satosi Watanabe. *Pattern Recognition: Human and Mechanical*. Wiley, Nueva York, NY, EE. UU., 1985. ISBN 978-0-471-80815-2.

[23] Yu Yi, Yinsheng Zhou y Ye Wang. A tempo-sensitive music search engine with multimodal inputs. En *Proceedings of the 1st International ACM Workshop on Music Information Retrieval with User-centered and Multimodal Strategies*, MIRUM '11, págs 13–18. ACM, Nueva York, NY, EE. UU., 2011. doi:10.1145/2072529.2072533.

[24] Tong Zhang, Chee Keat Fong, Linxing Xiao y Jie Zhou. Automatic and instant ring tone generation based on music structure analysis. En *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, págs 593–596. ACM, Nueva York, NY, EE. UU., 2009. doi:10.1145/1631272.1631364.

Ficha autobiográfica

Iván Alejandro Guerra López

Candidato para el grado de Ingeniero en Tecnología de Software

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

Por definir

Nacido en la ciudad de Monterrey, Nuevo León el 23 de septiembre de 1991. Segundo hijo de Jorge Arturo Guerra Aguilar y Martha Patricia López Arrambide. Cursando el bachillerato en la Preparatoria No. 15 unidad Madero de la Universidad Autonoma De Nuevo León. Empecé mis estudios universitarios en agosto del año 2009 en la Facultad de Ingeniería Mecánica y Eléctrica de la UANL en la carrera de Ingeniero en Tecnología de Software.